NUMERICALLY APPROXIMATING ZEROS OF A POLYNOMIAL

By

RANDY JOE SNIDER

Bachelor of Science

Oklahoma State University

Stillwater, Oklahoma

1968

Submitted to the Faculty of the Graduate School
of the Oklahoma State University
in partial fulfillment of the requirements
for the Degree of
MASTER OF SCIENCE
May, 1970

NUMERICALLY APPROXIMATING ZEROS OF A POLYNOMIAL

Thesis Approved:

Craig A. Wood
Thesis Adviser

Forrest D. Whitfield

Paul G. Miller

[illegible]
Dean of the Graduate College

PREFACE

In this study I have presented five iterative methods for approximating zeros of a polynomial using the digital computer. Chapter I is an introduction to the material which follows. Chapters II through VI contain the algorithms and convergence properties for Newton's, Muller's, Greatest Common Divisor, Lehmer's and the Quotient-Difference methods, respectively. Chapter VII compares Newton's, Muller's, and the Greatest Common Divisor methods, giving their advantages, disadvantages, and results of computer tests performed. Appendices B through E contain flow diagrams, program listings, and instructions for use of the programs for Newton's, Muller's, Greatest Common Divisor method used with Newton's method, and Greatest Common Divisor method used with Muller's method, respectively. Appendix A describes a number of special features performed by the programs. Appendix F presents flow diagrams, program listings, and instructions for use of a program to construct the polynomial resulting from the product of a given number of linear factors.

I would like to express my appreciation to Dr. Craig A. Wood for his assistance and guidance in directing the writing of this thesis.

Also I would like to thank the National Aeronautics and Space Administration for the financial support received as graduate assistant on research grant NASA NGR 37-002-084; to Dr. Wood for the opportunity of working with him on his grant; and to Dr. John W. Jewett, Head of the Department of Mathematics, for my teaching assistantship.

Furthermore, I would like to thank Dr. Forrest D. Whitfield and Dr. Paul G. Miller for serving as chairman and member, respectively, of my advisory committee.

In addition, I would like to thank Beverly Turner, Mathematics Departmental Secretary, for an excellent job done in typing the manuscript.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Figure Page

LIST OF EXHIBITS

CHAPTER I

INTRODUCTION

Frequently in scientific work it becomes necessary to find the zeros, real or complex, of the polynomial of degree N

$$P(X) = a_1X^N + a_2X^{N-1} + \dots + a_NX + a_{N+1}$$

where $a_1 \neq 0$, and the coefficients $a_1, a_2, \dots, a_{N+1}$ are complex numbers. Various calssical methods calculate the exact roots of polynomials of degree 1,2,3, or 4. For polynomials of higher degree, no such methods exist. Thus, to solve for the zeros of such polynomials, numerical methods of iteration based on successive approximations must be employed. In the following material five such methods are given which are particularly suited for modern high speed computers.

Newton's method is an iterative procedure which generates a sequence of successive approximations of a zero of P(X) by using the iteration formula

$$X_{n+1} = X_n - P(X_n)/P'(X_n).$$

An initial approximation to the zero is required to start the iterative process. Under certain conditions this sequence will converge quadratically to the desired root. It is, however, necessary to compute the value of the polynomial and its derivative for each step in the iterative procedure. Once a zero of P(X) has been found, it is divided out of P(X), giving a deflated polynomial of lower degree. P(X) is replaced by the deflated polynomial and the iterative process is applied to extract another zero of P(X). This procedure is repeated until all zeros of P(X) have been found. The zeros may then be rechecked and their accuracy possibly improved by using them as initial approximations with Newton's process applied to the full (undeflated) polynomial.

Muller's method is also an iterative procedure generating a sequence $X_1, X_2, \ldots, X_n, \ldots$ of successive approximations of a root of P(X). This method converges almost quadratically near a zero and does not require the evaluation of the derivative of the polynomial. Muller's method requires three distinct approximations of a root to start the process of iteration. A quadratic equation is constructed through the three given points as an approximation of P(X). The root of the quadratic closest to X_n is taken as X_{n+1}, the next approximation to the zero. This process is then repeated on the last three points of the sequence. After a root of P(X) has been found, P(X) is deflated, and replaced in the above procedure by the deflated polynomial. After all zeros of P(X) are found from successive deflations, they are improved as in Newton's method.

The greatest common divisor method reduces the problem of finding all zeros (possibly multiple zeros) of P(X) to one of extracting the zeros of a polynomial $P_1(X) = P(X)/D(X)$, all of whose zeros are simple. D(X), the greatest common divisor of P(X) and its derivative, $P'(X)$, is obtained by repeated application of the division algorithm. Once $P_1(X)$ is obtained, some suitable method such as Newton's or Muller's method is used to find the zeros of $P_1(X)$. By finding all the zeros of $P_1(X)$, all the zeros of $P(X)$ are obtained. The multiplicity of each zero may then be determined.

Lehmer's method locates zeros of the polynomial $P(X)$ by searching the complex plane. This method repeatedly seeks to answer the question "Does $P(X)$ have a zero inside a given circle?" The procedure is started with the unit circle. The radius is repeatedly halved (or doubled) until an annulus is found which contains a root while the inner circle is free of roots. The annulus can be covered by eight overlapping circles. The above question is asked of each circle until one is found containing a root. From the center of this circle the above process is repeated. Continuing, we obtain a circle of sufficiently small radius about the root. When a root has been found, it is divided out of $P(X)$ by deflation and the method repeated on the deflated polynomial to extract another root. Lehmer's method applied to the successively deflated polynomials gives all the roots of $P(X)$.

The quotient-difference method is an iterative procedure producing, simultaneously, approximations to all roots of $P(X)$. The zeros are extracted roughly in order of decreasing magnitude. No initial approximations are necessary for use of this method. Convergence of this method is somewhat slower than other methods and substantial round-off error can occur.

In the material to follow, an individual examination of each method will be presented followed by a conclusion containing a comparison of the methods. These examinations will include the algorithms necessary to employ the method together with the conditions necessary for convergence. The conclusion will include advantages and disadvantages of each method and a discussion concerning the extraction of zeros of ill-conditioned polynomials. An appendix for each method will present the flow diagrams, listing of the program, definition of variables used in the program, and instructions for use of the program.

CHAPTER II

NEWTON'S METHOD

1. Derivation of the Algorithm

Newton's method is probably the most popular iterative procedure for finding the zeros of a polynomial. This fact is due to the excellent results obtained, the simplicity of the computational routine, and the fast rate of convergence obtained provided the initial approximation of a zero is close enough. Also, the method can be applied to the extraction of complex as well as real zeros.

Consider the polynomial

$$P(X) = a_1X^N + a_2X^{N-1} + \ldots + a_NX + a_{N+1} \qquad (2\text{-}1)$$

where $a_1 \neq 0$ and the coefficients $a_1, a_2, \ldots, a_{N+1}$ are complex. The algorithm for Newton's method can be derived by approximating P(X) by a Taylor series expansion about an approximation, X_0, of a zero, α, of P(X). Using only the first two terms of the expansion, the expression

$$P(X) \doteq P(X_0) + P'(X_0)(X - X_0)$$

is obtained. If this equation is solved for P(X) = 0, then

$$0 \doteq P(X_0) + P'(X_0)(X - X_0)$$

results. Rearranging terms produces

$$0 \doteq P(X_0) + P'(X_0)\, X - P'(X_0)\, X_0$$

followed by

$$P'(X_0)\, X_0 - P(X_0) \doteq P'(X_0)\, X$$

from which division by $P'(X_0)$ produces

$$X_0 - P(X_0)/P'(X_0) \doteq X$$

which is the basic formula for Newton's method. Thus, in general, we obtain the (n+1)th approximation, X_{n+1}, of α from the nth approximation, X_n, by

$$X_{n+1} = X_n - P(X_n)/P'(X_n). \qquad (2\text{-}2)$$

As a result of repeated use of this algorithm, we obtain the sequence

$$X_0, X_1, X_2, \ldots, X_n, \ldots \qquad (2\text{-}3)$$

of successive approximations of the root, α. It should be noted that an initial approximation is necessary to start the iterative process for each new zero; that is, a polynomial of degree N may require N initial approximations.

In order to use equation (2-2), it is necessary to compute, for each X_n, the value of the polynomial, $P(X_n)$, and its derivative, $P'(X_n)$. The division algorithm states that if $P(X)$ and $G(X)$ are polynomials, then there exists polynomials $H(X)$ and $K(X)$ such that $P(X) = H(X)\, G(X) + K(X)$ where $K(X) = 0$ or deg. $K(X) <$ deg. $G(X)$. From this expression of $P(X)$, the following remainder theorem is obtained:

Theorem 2.1. If $P(X)$ is a polynomial and c is a complex number, then the remainder obtained from dividing $P(X)$ by $(X - c)$ is $P(c)$.

The proof of Theorem 2.1 is given in [6, P. 102]. Thus, $P(X)$ can be written as $P(X) = (X - c)\,H(X) + R$ where $P(c) = R$. $P'(X)$ is then obtained by the following theorem, the proof of which can be found in [6, PP. 105-106].

Theorem 2.2. If $P(X)$ and $H(X)$ are polynomials and c is a complex number such that $P(X) = (X - c)\,H(X) + R$ where $P(c) = R$, then the remainder obtained from dividing $H(X)$ by $(X - c)$ is $P'(c)$.

From synthetic division, an algorithm known as Horner's Method is acquired for computing $P(X_n)$ and $P'(X_n)$.

Theorem 2.3. Let $P(X)$ be defined as in equation (2-1) and let d be a complex number. Define a sequence $b_1, b_2, \ldots, b_{N+1}$ by

$$b_1 = a_1$$

$$b_i = a_i + db_{i-1} \quad (i = 2,3,\ldots,N+1).$$

Define another sequence $c_1, c_2, \ldots, c_N$ by

$$c_1 = b_1$$

$$c_j = b_j + dc_{j-1} \quad (j = 2,3,\ldots,N).$$

Then $P(d) = b_{N+1}$ and $P'(d) = c_N$. The elements $b_1, b_2, \ldots, b_N$ are the coefficients of the polynomial $H(X)$ in Theorem 2.2 when $P(X)$ is divided by $(X - d)$.

These formulas are derived in [6, PP. 106-107]. Thus with equation (2-2) and the iteration formulas of the previous theorem, Newton's method can now be applied to generate the sequence (2-3) which will converge to the root, α, if the convergence conditions given in Theorem 2.4 are satisfied.

A criterion is needed to determine when to terminate the sequence (2-3); that is, when has a zero been found? For convergence of the sequence, there must exist a term in the sequence beyond which the difference between any two successive terms is arbitrarily small. Therefore, it is desirable to make the quotient $|X_n/X_{n+1}|$ sufficiently near 1. From equation (2-2)

$$1 = \left| \frac{X_n}{X_{n+1}} - \frac{\frac{P(X_n)}{P'(X_n)}}{X_{n+1}} \right|$$

$$\leq \left| \frac{X_n}{X_{n+1}} \right| - \left| \frac{\frac{P(X_n)}{P'(X_n)}}{X_{n+1}} \right| .$$

Thus

$$1 + \frac{\left|\frac{P(X_n)}{P'(X_n)}\right|}{|X_{n+1}|} \leq \left| \frac{X_n}{X_{n+1}} \right|$$

where $P'(X_n)$ and $X_{n+1} \neq 0$. Thus, iterations are continued until an X_n is obtained such that $|P(X_n)/P'(X_n)| / |X_{n+1}|$ is as small as desired.

After a zero, α, of $P(X)$ has been found, the term $(X - \alpha)$ is synthetically divided out of $P(X)$ by deflation using Theorem 2.3 obtaining a polynomial, $P_1(X)$, of degree N-1. The root finding process is then repeated to extract a zero, α_1, of $P_1(X)$. P(X) can be written as

$$P(X) = (X - \alpha)\ P_1(X) + R$$

where $R = P(\alpha)$. But $P(\alpha) = 0$. Therefore, substitution produces

$$P(X) = (X - \alpha)\ P_1(X).$$

Now $P_1(\alpha_1) = 0$ implies that $P(\alpha_1) = 0$. Hence, α_1 is a zero of P(X).

By the process of root finding and successive deflations, zeros $\alpha_0, \alpha_1, \ldots, \alpha_{N-1}$ of the deflated polynomials

$$P(X) = P_0(X), P_1(X), \ldots, P_{N-1}(X),$$

respectively, are extracted. Each α_i $(i = 0,1,2,\ldots,N-1)$ is a zero of P(X) since each α_i is a zero of $P_{i-1}(X), P_{i-2}(X), \ldots, P_1(X), P(X)$.

After all zeros of P(X) have been found, it may be possible to improve their accuracy by using them as initial approximations with Newton's method applied to the full (undeflated) polynomial, P(X). This should correct any loss of accuracy which may have resulted from the successive deflations.

2. Convergence of Newton's Method

The following theorem from [3, PP. 79-81] gives sufficient conditions for the convergence of sequence (2-3).

Theorem 2.4. Let P(X) be a polynomial and let the following conditions be satisfied on the closed interval [a,b]:

1. $P(a)\,P(b) < 0$
2. $P'(X) \neq 0,\ X \in [a,b]$
3. $P''(X)$ is either ≥ 0 or ≤ 0 for all $X \in [a,b]$
4. If c denotes the endpoint of $[a,b]$ at which $|P'(X)|$ is smaller, then $|P(c)/P'(c)| \leq b - a$.

Then Newton's method converges to the (only) solution, s, of $P(X) = 0$ for any choice of X_0 in $[a,b]$.

When convergence is obtained, it is quadratic; that is,

$$e_{i+1} = \frac{1}{2} F''(\eta_i)\, e_i^2$$

where $F(X_i) = X_i - P(X_i)/P'(X_i)$, η_i is between X_i and the zero, α, and e_i is the error in X_i. This means that the error obtained in the (i+1)th iteration of Newton's algorithm is proportional to the square of the error obtained in the ith iteration. A proof of quadratic convergence can be found in [2, PP. 31-33].

3. Procedure for Newton's Method

The general procedure for applying Newton's method is enumerated sequentially as follows, starting with initial approximation X_0:

1. Calculate a new approximation X_{n+1} by
$$X_{n+1} = X_n - P(X_n)/P'(X_n).$$
2. Test for convergence; that is, test
$$|P(X_n)/P'(X_n)|/|X_{n+1}| < \varepsilon$$
for some ε chosen as small as desired.
3. If convergence is obtained, perform the following:

a. Save X_{n+1} as the desired approximation to a zero of P(X).

b. Deflate P(X) using X_{n+1}.

c. Replace P(X) by the deflated polynomial.

d. Return to step 1 with a new initial approximation.

4. If no convergence is obtained, increase n by 1 and return to step 1.

In order to prevent an unending iteration process in case the method does not produce convergence, a maximum number of iterations should be specified. If convergence is not obtained within this number of iterations, change the initial approximation and return to step 1 above.

4. Geometrical Interpretation of Newton's Method

A geometrical interpretation of Newton's method is given in Figure 1. X_i is an approximation to the zero, α. $P'(X_i)$ is the slope of the line tangent to P(X) at X_i. X_{i+1} is the intersection of the tangent line with the x axis.

5. Determining Multiple Roots

If P(X) has m distinct zeros, then P(X) can be written as

$$P(X) = a_1(X - \alpha_1)^{e_1} (X - \alpha_2)^{e_2} \ldots (X - \alpha_m)^{e_m}, \quad (m \leq N)$$

where α_i is a zero of P(X) and e_i is the multiplicity of α_1 $(i = 1,2,\ldots,m)$. Consider the root α_j. Dividing out the term

$(X - \alpha_j)$ by deflating $P(X)$ gives $P_1(X)$ of degree N-1 which can be written as

$$P_1(X) = (X - \alpha_1)^{e_1} (X - \alpha_2)^{e_2} \dots (X - \alpha_j)^{e_j - 1} \dots (X - \alpha_m)^{e_m}.$$

Evaluating $P_1(X)$ at the zero, α_j, gives $P_1(\alpha_j) = 0$ if $e_j > 1$. Thus, after a zero, α, of $P(X)$ is determined by Newton's iterative process and the current polynomial is deflated giving $P_1(X)$, then $P_1(\alpha)$ is evaluated. If $P_1(\alpha) \leq \varepsilon$ for some small number ε, α is a root of $P_1(X)$ and thus has multiplicity at least equal to two. $P_1(X)$ is then deflated giving $P_2(X)$. If $P_2(\alpha) \leq \varepsilon$, α is of multiplicity at least three. This process is continued until a deflated polynomial $P_k(X)$ is encountered such that either deg. $P_k(X) = 0$ or $P_k(\alpha) > \varepsilon$. α is then a zero of multiplicity k+1.

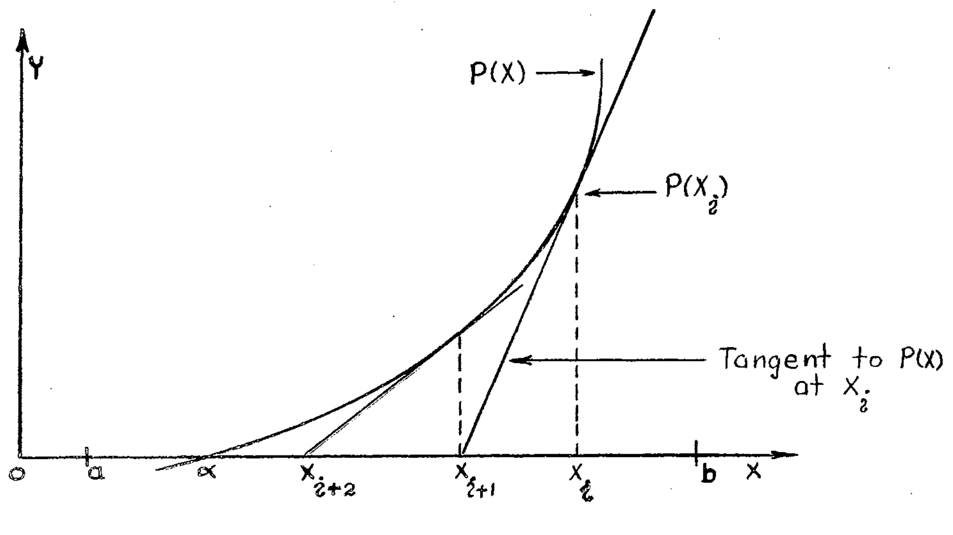


Figure 1. Geometrical Interpretation of Newton's Method

CHAPTER III

MULLER'S METHOD

1. Derivation of the Algorithm

Muller's method in [5] is an iterative procedure designed to find any prescribed number of zeros, real or complex, of a polynomial. The method does not require the evaluation of the derivative and near a zero the convergence is almost quadratic.

Consider the polynomial

$$P(X) = a_1X^N + a_2X^{N-1} + \dots + a_NX + a_{N+1} \tag{3-1}$$

with complex coefficients such that $a_1 \neq 0$. Given three distinct approximations, X_{n-2}, X_{n-1}, X_n, to a root, α, of $P(X)$, the problem is to determine X_{n+1} in such a way as to generate a sequence

$$X_1, X_2, X_3, \dots, X_n, X_{n+1}, \dots \tag{3-2}$$

of approximations converging to α. The points $(X_{n-2}, P(X_{n-2}))$, $(X_{n-1}, P(X_{n-1}))$, and $(X_n, P(X_n))$ determine a unique quadratic polynomial, $Q(X)$, approximating $P(X)$ in the vicinity of X_{n-2}, X_{n-1}, X_n. A general proof of this can be found in [3, PP. 133-134]. Thus, the zeros of $Q(X)$ will be approximations of the zeros of $P(X)$ in this region of approximation. From the general representation in [3, P. 184] of the Legrangian interpolating polynomial, the representation of $Q(X)$ is given by

$$Q(X) = \frac{(X - X_{n-1})(X - X_{n-2})}{(X_n - X_{n-1})(X_n - X_{n-2})} P(X_n)$$

$$+ \frac{(X - X_n)(X - X_{n-2})}{(X_{n-1} - X_n)(X_{n-1} - X_{n-2})} P(X_{n-1})$$

$$+ \frac{(X - X_n)(X - X_{n-1})}{(X_{n-2} - X_n)(X_{n-2} - X_{n-1})} P(X_{n-2})$$

which can be rewritten as

$$Q(X) = Q(X - X_n + X_n)$$

$$= \frac{(X - X_n + X_n - X_{n-1})(X - X_n + X_n - X_{n-1} + X_{n-1} - X_{n-2})}{(X_n - X_{n-1})(X_n - X_{n-1} + X_{n-1} - X_{n-2})} P(X_n)$$

$$- \frac{(X - X_n)(X - X_n + X_n - X_{n-1} + X_{n-1} - X_{n-2})}{(X_n - X_{n-1})(X_{n-1} - X_{n-2})} P(X_{n-1})$$

$$+ \frac{(X - X_n)(X - X_n + X_n - X_{n-1})}{(X_n - X_{n-1} + X_{n-1} - X_{n-2})(X_{n-1} - X_{n-2})} P(X_{n-2}).$$

In order to simplify this expression, introduce the quantities

$$h_n = X_n - X_{n-1}, \quad h = X - X_n.$$

Then

$$Q(X) = Q(X_n + h)$$

$$= \frac{(h + h_n)(h + h_n + h_{n-1})}{h_n(h_n + h_{n-1})} P(X_n)$$

$$- \frac{h(h + h_n + h_{n-1})}{h_n h_{n-1}} P(X_{n-1})$$

$$+ \frac{h(h + h_n)}{(h_n + h_{n-1})h_{n-1}} P(X_{n-2})$$

$$= \frac{h^2 + 2hh_n + hh_{n-1} + h_n^2 + h_n h_{n-1}}{h_n^2 + h_n h_{n-1}} P(X_n)$$

$$- \frac{h^2 + hh_n + hh_{n-1}}{h_n h_{n-1}} P(X_{n-1})$$

$$+ \frac{h^2 + hh_n}{h_n h_{n-1} + h_{n-1}^2} P(X_{n-2}).$$

Collecting terms containing like powers of h produces

$$Q(X) = Q(X_n + h)$$

$$= \left(\frac{P(X_n)}{h_n^2 + h_n h_{n-1}} - \frac{P(X_{n-1})}{h_n h_{n-1}} + \frac{P(X_{n-2})}{h_n h_{n-1} + h_{n-1}^2} \right) h^2$$

$$+ \left(\frac{(2h_n + h_{n-1})\ P(X_n)}{h_n^2 + h_n h_{n-1}} - \frac{(h_n + h_{n-1})\ P(X_{n-1})}{h_n h_{n-1}} + \frac{h_n\ P(X_{n-2})}{h_n h_{n-1} + h_{n-1}^2} \right) h$$

$$+ \frac{h_n (h_n + h_{n-1})\ P(X_n)}{h_n^2 + h_n h_{n-1}}$$

$$= \left(\frac{P(X_n)\ h_{n-1}}{h_n^2 h_{n-1} + h_n h_{n-1}^2} - \frac{P(X_{n-1})}{h_n h_{n-1}} + \frac{P(X_{n-2})}{h_n h_{n-1} + h_{n-1}^2} \right) h^2$$

$$+ \left(\frac{(2h_n h_{n-1} + h_{n-1}^2)\ P(X_n)}{h_n^2 h_{n-1} + h_n h_{n-1}^2} - \frac{(h_n + h_{n-1}) P(X_{n-1})}{h_n h_{n-1}} + \frac{h_n\ P(X_{n-2})}{h_n h_{n-1} + h_{n-1}^2} \right) h$$

$$+ \frac{(h_n^2 h_{n-1} + h_{n-1}^2 h_n)\, P(X_n)}{h_n^2 h_{n-1} + h_n h_{n-1}^2} \;.$$

Using the common denominator, $h_n^2 h_{n-1} + h_n h_{n-1}^2$, and combining terms yields

$$Q(X_n + h) = \left(\frac{P(X_n)\, h_{n-1} - P(X_{n-1})(h_n + h_{n-1}) + P(X_{n-2})\, h_n}{h_n^2 h_{n-1} + h_n h_{n-1}^2}\right) h^2$$

$$+ \left(\frac{(2h_n h_{n-1} + h_{n-1}^2)\, P(X_n) - (h_n + h_{n-1})^2\, P(X_{n-1}) + h_n^2 P(X_{n-2})}{h_n^2 h_{n-1} + h_n h_{n-1}^2}\right) h$$

$$+ \frac{(h_n^2 h_{n-1} + h_{n-1}^2 h_n)\, P(X_n)}{h_n^2 h_{n-1} + h_n h_{n-1}^2} \;.$$

Multiplying by h_n/h_{n-1}^2 results in

$$Q(X_n + h) = \left[\frac{P(X_n)\, \dfrac{h_n}{h_{n-1}} - P(X_{n-1}) \left(\left(\dfrac{h_n}{h_{n-1}}\right)^2 + \dfrac{h_n}{h_{n-1}}\right) + P(X_{n-2}) \left(\dfrac{h_n}{h_{n-1}}\right)^2}{\dfrac{h_n^3}{h_{n-1}} + h_n^2}\right] h^2$$

$$+ \left[\frac{\left(2\dfrac{h_n^2}{h_{n-1}} + h_n\right) P(X_n) - h_n \left[\left(\dfrac{h_n}{h_{n-1}}\right) + \left(\dfrac{h_{n-1}}{h_{n-1}}\right)\right]^2 P(X_{n-1}) + \dfrac{h_n^3}{h_{n-1}^2}\, P(X_{n-2})}{\dfrac{h_n^3}{h_{n-1}} + h_n^2}\right] h$$

$$+ \left[\frac{\left(\frac{h_n^3}{h_{n-1}} + h_n^2 \right) P(X_n)}{\frac{h_n^3}{h_{n-1}} + h_n^2} \right] .$$

Let $q_n = \frac{h_n}{h_{n-1}}$ and $q = \frac{h}{h_n}$. Then

$$Q(X_n + h) = \left(\frac{P(X_n)\, q_n - P(X_{n-1})(q_n^2 + q_n) + P(X_{n-2})\, q_n^2}{q_n+1} \right) q^2$$

$$+ \left(\frac{(2\, q_n+1)\, P(X_n) - (q_n+1)^2\, P(X_{n-1}) + q_n^2\, P(X_{n-2})}{q_n+1} \right) q$$

$$+ \frac{(q_n+1)\, P(X_n)}{q_n+1} .$$

Now let

$$A_n = q_n\, P(X_n) - q_n(q_n+1)\, P(X_{n-1}) + q_n^2\, P(X_{n-2})$$

$$B_n = (2q_n+1)\, P(X_n) - (q_n+1)^2\, P(X_{n-1}) + q_n^2\, P(X_{n-2})$$

$$C_n = (q_n+1)\, P(X_n).$$

Then

$$Q(X_n + h) = Q(X_n + qh_n)$$

and

$$Q(X_n + qh_n) = \frac{A_n\, q^2 + B_n\, q + C_n}{q_n+1} .$$

Solving the quadratic equation $Q(X_n + qh_n) = 0$ and denoting the result by q_{n+1} gives:

$$q_{n+1} = \frac{-B_n \pm \sqrt{B_n^2 - 4A_nC_n}}{2A_n}$$

and the new approximation is found as follows:

$$q_{n+1} = \frac{h_{n+1}}{h_n} = \frac{X_{n+1} - X_n}{h_n} .$$

Thus

$$X_{n+1} = X_n + h_n \, q_n+1.$$

In order to avoid loss of accuracy, q_{n+1} can be written in a better form as follows:

$$q_{n+1} = \frac{-B_n \pm \sqrt{B_n^2 - 4A_nC_n}}{2A_n} \cdot \frac{B_n \pm \sqrt{B_n^2 - 4A_nC_n}}{B_n \pm \sqrt{B_n^2 - 4A_nC_n}}$$

$$= \frac{-B_n^2 + B_n^2 - 4A_nC_n}{2A_n \ (B_n \pm \sqrt{B_n^2 - 4A_nC_n})}$$

$$q_{n+1} = \frac{-2C_n}{B_n \pm \sqrt{B_n^2 - 4A_nC_n}} . \tag{3-3}$$

The sign in the denominator should be chosen such that the magnitude of the denominator is largest, thus causing $|q_{n+1}|$ to be smallest. This, in turn, will make X_{n+1} closest to X_n.

Note that each iteration of this process requires three approximations, X_{n-2}, X_{n-1}, X_n, in order to compute X_{n+1}. Thus, when X_{n+1} is found, X_{n-1}, X_n, X_{n+1} are used to compute X_{n+2}; that is, the last three terms of the generated sequence are used to compute the next term.

Convergence of the sequence (3-2) to a zero is obtained when the elements X_k and X_{k+1} of the sequence are found such that

$$\frac{|X_{k+1} - X_k|}{|X_{k+1}|} < \varepsilon, \; X_{k+1} \neq 0;$$

that is, the ratio of the change in the approximation to the approximation itself is as small as desired.

In order to use the iterative formulas, it is necessary to compute the value, $P(X_j)$, of the polynomial $P(X)$ at the approximation X_j. The procedure for doing this is discussed in Chapter II, § 1. The iteration formulas are given in Theorem 2.3 of Chapter II.

After a zero, α, of $P(X)$ has been found, $P(X)$ is deflated as described in Chapter II, § 1, and the process repeated to extract a zero, α_1, of $P_1(X)$. By applying Muller's method to successively deflated polynomials, all the zeros of $P(X)$ are obtained. For more detailed discussion of this procedure see Chapter II, § 1, keeping in mind that Muller's instead of Newton's method is used.

Muller's method requires three initial approximations to a zero in order to start the iteration process. If three are not known, the values $X_1 = -1$, $X_2 = 1$, $X_3 = 0$ can be used.

Convergence of Muller's method is almost quadratic provided the three initial approximations are sufficiently close to a zero of $P(X)$. This is natural to expect since $P(X)$ is being approximated by a quadratic polynomial. Quadratic convergence means that the error obtained in the $(n+1)^{th}$ step of the iterative process is proportional to the square of the error obtained in the n^{th} iteration. However, no general proof of convergence has been obtained for Muller's method. It has produced convergence in the majority of the cases tested.

In application of Muller's method, an alteration should be made to handle the case in which the denominator of equation (3-3) is zero (0). This occurs whenever $P(X_n) = P(X_{n-1}) = P(X_{n-2})$. If this happens, set $q_{n+1} = 1$.

Another alteration which should be made in actual practice is to compute the quantity $|P(X_{n+1})| / |P(X_n)|$ whenever the value $P(X_{n+1})$ is calculated. If the former quantity exceeds ten (10), q_{n+1} is halved and h_n, X_{n+1}, and $P(X_{n+1})$ are recomputed accordingly.

2. Procedure for Muller's Method

The basic steps performed by Muller's method are listed sequentially as follows, starting with initial approximations X_1, X_2, and X_3.

1. Compute h_n, q_n, D_n, B_n, C_n, q_{n+1} as defined previously.
2. Compute the next approximation X_{n+1} by
$$X_{n+1} = X_n + h_n\, q_{n+1}.$$
3. Test for convergence; that is, test
$$|X_{n+1} - X_n| / |X_{n+1}| < \varepsilon$$
for some suitably small number ε.
4. If the test fails, return to step 1 with the last three approximations X_{n+1}, X_n, X_{n-1}.

5. If the test passes, do the following:
   a. Save X_{n+1} as the desired approximation to a zero.
   b. Deflate the current polynomial using X_{n+1}.
   c. Replace the current polynomial by the deflated polynomial.
   d. Return to step 1 with a new set of initial approximations.

In order to avoid an unending iteration process in case the method does not produce convergence, a maximum number of iterations should be specified. If convergence is not obtained within this number of iterations, the initial approximations should be altered.

3. Geometrical Interpretation of Muller's Method

Figure 2 shows the geometrical interpretation of Muller's method for real roots of P(X) and the quadratic Q(X). The root of Q(X) closest to X_i is chosen as the next approximation X_{i+1}.

4. Determining Multiple Roots

For a discussion concerning multiple roots see Chapter II, § 5.

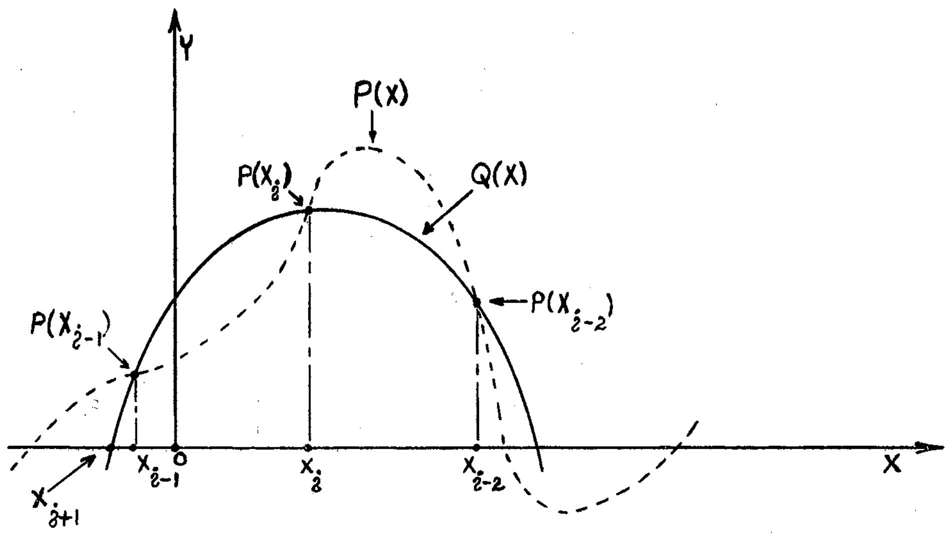


Figure 2. Geometrical Interpretation of Müller's Method

CHAPTER IV

GREATEST COMMON DIVISOR METHOD

1. Derivation of the Algorithm

The greatest common divisor (g.c.d.) method reduces the problem of finding all the zeros of a polynomial, possibly having multiple zeros, to one of solving for zeros of a polynomial all of whose zeros are simple.

Consider the N^{th} degree polynomial

$$P(X) = a_1X^N + a_2X^{N-1} + \dots + a_NX + a_{N+1} \tag{4-1}$$

where $a_1 \neq 0$ and $a_1, a_2, \dots, a_{N+1}$ are complex numbers. If P(X) has m distinct zeros, $\alpha_1, \alpha_2, \dots, \alpha_m$, then P(X) can be expressed in the form

$$P(X) = a_1(X - \alpha_1)^{e_1} (X - \alpha_2)^{e_2} \dots (X - \alpha_m)^{e_m}$$

where e_i is the multiplicity of α_i, $i = 1,2,\dots,m$. The derivative of P(X) is

$$P'(X) = N\, a_1X^{N-1} + (N-1)\, a_2X^{N-2} + \dots + a_N$$

which can also be expressed as

$$P'(X) = a_1(X - \alpha_1)^{e_1-1} (X - \alpha_2)^{e_2-1} \dots (X - \alpha_m)^{e_m-1} \sum_{i=1}^{m} e_i \prod_{\substack{j=1 \\ j\neq i}}^{m} (X - \alpha_j). \tag{4-2}$$

The greatest common divisor of P(X) and P'(X) is obtained from the following theorem.

Theorem 4.1. Let P(X) be an N^{th} degree polynomial having m distinct zeros $\alpha_1, \alpha_2, \dots, \alpha_m$ of multiplicity $e_1, e_2, \dots, e_m$ respectively. Then the polynomial

$$D(X) = (X - \alpha_1)^{e_1-1} (X - \alpha_2)^{e_2-1} \dots (X - \alpha_m)^{e_m-1}$$

is the unique monic greatest common divisor of P(X) and its derivative P'(X).

Proof. It suffices to show the following:

a. D(X) divides P(X)

b. D(X) divides P'(X)

c. If K(X) is a polynomial such that K(X) divides both P(X) and P'(X), then K(X) divides D(X).

P'(X) can be written as

$$P'(X) = D(X)\ S(X) \text{ where } S(X) = \sum_{i=1}^{m} e_i \prod_{\substack{j=1 \\ j\neq i}}^{m} (X - \alpha_j).$$

D(X) divides P(X) since

$$P(X) = D(X) \prod_{i=1}^{m} (X - \alpha_i).$$

D(X) divides P'(X) because

$$P'(X) = D(X) \sum_{i=1}^{m} e_i \prod_{\substack{j=1 \\ j\neq i}}^{m} (X - \alpha_j).$$

Both the polynomials P(X) and P'(X) can be expressed uniquely as a constant times a product of monic irreducible polynomials. A general proof of this can be found in [7, P. 121]. Note that even though there are e_i of the factors $(X - \alpha_i)$ in P(X) and e_i-1 of them in P'(X), we will consider them distinct in the sense that they are not identical. Let K(X) be a non-constant polynomial such that K(X) divides P(X) and P'(X). Without loss of generality we may assume that K(X) is monic. Since K(X) divides P(X), then there exists a polynomial T(X) such that

$$P(X) = K(X)\ T(X). \tag{4-3}$$

From Theorem 3.35 of [7, P. 121], K(X) and T(X) can be expressed in the forms:

$$K(X) = (X - \beta_1)^{g_1} (X - \beta_2)^{g_2} \ldots (X - \beta_v)^{g_v}$$

$$T(X) = b(X - \eta_1)^{f_1} (X - \eta_2)^{f_2} \ldots (X - \eta_u)^{f_u}$$

where the factors are unique apart from order. Substituting for P(X), T(X), and K(X) in equation (4-3) gives

$$a_1(X - \alpha_1)^{e_1} (X - \alpha_2)^{e_2} \dots (X - \alpha_m)^{e_m}$$

$$= [(X - \beta_1)^{g_1} \dots (X - \beta_v)^{g_v}] [b(X - \eta_1)^{f_1} \dots (X - \eta_u)^{f_u}]; \qquad (4\text{-}4)$$

that is, both sides of equation (4-4) are unique representations of P(X). After proper ordering of the terms on the right hand side of equation (4-4), we conclude from the uniqueness of the two expressions that each factor of K(X) must be a factor of P(X). This implies that $X - \beta_i = X - \alpha_i$ for $i = 1,\dots,v$. Also, $g_i \leq e_i$ for $i = 1,2,\dots,v$ since K(X) cannot contain a greater number of any one factor than P(X). Finally, $v \leq m$ since K(X) cannot have more distinct factors than P(X). Thus

$$K(X) = (X - \alpha_1)^{g_1} (X - \alpha_2)^{g_2} \dots (X - \alpha_v)^{g_v}. \qquad (4\text{-}5)$$

Since K(X) divides $P'(X)$, there exists a polynomial R(X) such that

$$P'(X) = K(X)\,R(X). \qquad (4\text{-}6)$$

Again, R(X) can be expressed uniquely as

$$R(X) = d(X - \lambda_1)^{h_1} (X - \lambda_2)^{h_2} \dots (X - \lambda_t)^{h_t}.$$

Also, $P'(X)$, given in equation (4-2), can be expressed as

$$P'(X) = [a_1(X - \alpha_1)^{e_1 - 1} (X - \alpha_2)^{e_2 - 1} \dots (X - \alpha_m)^{e_m - 1}]$$

$$[c(X - \delta_1)^{e_{m+1}} \dots (X - \delta_w)^{e_{m+w}}]$$

since

$$\sum_{i=1}^{m} e_i \prod_{\substack{j=1 \\ j\neq i}}^{m} (X - \alpha_j)$$

is a polynomial and can be expressed as a constant times a product of unique linear factors. Thus, substituting for $P'(X)$, $K(X)$, and $R(X)$ of equation (4-6) gives

$$[a_1(X - \alpha_1)^{e_1-1} \ldots (X - \alpha_m)^{e_m-1}]\,[c(X - \delta_1)^{e_{m+1}} \ldots (X - \delta_w)^{e_{m+w}}]$$

$$= [(X - \alpha_1)^{g_1} \ldots (X - \alpha_v)^{g_v}]\,[d(X - \lambda_1)^{h_1} \ldots (X - \lambda_t)^{h_t}].$$

Again, since both sides are unique representations of $P'(X)$, then each factor $X - \alpha_i$ must be contained in $P'(X)$ for $i = 1,\ldots,v$. But $X - \alpha_i$ is not in the product

$$\prod_{\substack{j=1 \\ j\neq i}}^{m} (X - \alpha_j)$$

and is, therefore, not a factor of $c(X - \delta_1)^{e_{m+1}} \ldots (X - \delta_w)^{e_{m+w}}$. Hence, $X - \alpha_1$ is contained in $a_1(X - \alpha_1)^{e_1-1} \ldots (X - \alpha_m)^{e_m-1}$ for $i = 1,\ldots,v$. Thus, $g_i \leq e_i-1$, $i = 1,2,\ldots,v$ since $K(X)$ cannot have a greater number of any one factor than $a_1(X - \alpha_1)^{e_1-1} \ldots (X - \alpha_m)^{e_m-1}$. Thus, $g_i \leq e_i-1$, $i = 1,\ldots,v$ which implies that each factor of $K(X)$ is contained in the factorization of $D(X)$. Hence, $K(X)$ divides $D(X)$. Therefore, $D(X)$ is the monic greatest common divisor of $P(X)$ and $P'(X)$.

$D(X)$ is unique since if $\overline{D}(X)$ is a monic g.c.d. of $P(X)$ and $P'(X)$, then $\overline{D}(X)$ divides $P(X)$ and $P'(X)$ and hence $D(X)$. But $D(X)$ divides both $P(X)$ and $P'(X)$ and hence $\overline{D}(X)$. Thus, $D(X) = \overline{D}(X)$ since both are monic polynomials.

Consider the polynomial $H(X)$ obtained by dividing $P(X)$ by its monic g.c.d., $D(X)$.

$$\begin{aligned} H(X) &= P(X)/D(X) \\ &= a_1 \prod_{i=1}^{m} (X - \alpha_i)^{e_i} \Big/ \prod_{i=1}^{m} (X - \alpha_i)^{e_i - 1} \\ &= a_1 \prod_{i=1}^{m} (X - \alpha_i). \end{aligned}$$

The zeros of $H(X)$ are all simple zeros and are also all the distinct zeros of $P(X)$. Use of the g.c.d. method involves computation of $H(X)$ when given $P(X)$.

In order to obtain $H(X)$, a computational algorithm is necessary to find the g.c.d. of $P(X)$ and $P'(X)$. The general method for computing the g.c.d. of two polynomials is as follows: Let $R_0(X)$ and $R_1(X)$ be two polynomials having degrees N_0 and N_1 respectively such that $N_1 \leq N_0$. The g.c.d. of $R_0(X)$ and $R_1(X)$ is desired. By the division algorithm, there exists polynomials $S_1(X)$ and $R_2(X)$ such that $R_0(X) = R_1(X)\, S_1(X) + R_2(X)$ where either $R_2(X) = 0$ or deg. $R_2(X) <$ deg. $R_1(X)$. Similarly if $R_2(X) \neq 0$, there exists polynomials $S_2(X)$ and $R_3(X)$ such that $R_1(X) = S_2(X)\, R_2(X) + R_3(X)$ where either $R_3(X) = 0$ or deg. $R_3(X) <$ deg. $R_2(X)$. Continuing in the above manner, suppose $R_i(X)$ and $R_{i+1}(X)$ have been found where deg. $R_{i+1}(X)$ < deg. $R_i(X)$. Then there exists polynomials $R_{i+2}(X)$ and $S_{i+1}(X)$ such that $R_i(X) = R_{i+1}(X)\ S_{i+1}(X) + R_{i+2}(X)$ where either $R_{i+2}(X) = 0$ or deg. $R_{i+2}(X)$ < deg. $R_{i+1}(X)$. Then we obtain a sequence $R_0(X), R_1(X), \ldots, R_K(X), R_{K+1}(X)$ such that deg. $R_i(X)$ < deg. $R_{i-1}(X)$, $i = 1,2,\ldots,K+1$. Since a polynomial cannot have degree less than zero, the above process, in a finite number of steps (at most N_1), results in polynomials $R_{K-1}(X)$, $S_K(X)$ and $R_K(X)$ with deg. $R_K(X)$ < deg. $R_{K-1}(X)$ such that $R_{K-1}(X) = R_K(X)\ S_K(X) + R_{K+1}(X)$ and $R_{K+1}(X) = 0$.

As an example, consider the proglem of finding the g.c.d. of $R_1(X) = X^3-1$ and $R_0(X) = X^4 + X^3 + 2X^2 + 1$. Then

$$R_0(X) = R_1(X)\ (X+1) + 2X^2 + 2X + 2$$

where $R_2(X) = 2X^2 + 2X + 2$. But

$$R_1(X) = R_2(X)\ (\tfrac{1}{2}X - \tfrac{1}{2}) + 0$$

where $R_3(X) = 0$. This theory leads to the following theorem.

<u>Theorem 4.2</u>. Let the sequence $R_0(X), R_1(X), \ldots, R_K(X), R_{K+1}(X)$ be defined as above. Then $R_K(X)$ is the greatest common divisor of $R_0(X)$ and $R_1(X)$.

<u>Proof</u>. Since $R_{K-1}(X) = R_K(X)\ S_K(X) + R_{K+1}(X)$ where $R_{K+1}(X) = 0$, then $R_{K-1}(X) = R_K(X)\ S_K(X)$ and thus $R_K(X)$ divides $R_{K-1}(X)$. Also

$$\begin{aligned} R_{K-2}(X) &= R_{K-1}(X)\ S_{K-1}(X) + R_K(X) \\ &= R_K(X)\ S_K(X)\ S_{K-1}(X) + R_K(X) \end{aligned}$$

$$= R_K(X)\ [S_K(X)\ S_{K-1}(X) + 1].$$

Thus, $R_K(X)$ divides $R_{K-2}(X)$. Suppose $R_K(X)$ divides $R_i(X)$ for $0 \leq i < i+1 \leq K$. This produces

$$R_i(X) = R_K(X)\ F_i(X) \quad \text{and}$$

$$R_{i+1}(X) = R_K(X)\ F_{i+1}(X)$$

where $F_i(X)$ and $F_{i+1}(X)$ are polynomials. Then

$$\begin{aligned} R_{i-1}(X) &= R_i(X)\ S_i(X) + R_{i+1}(X) \\ &= [R_K(X)\ F_i(X)]\ S_i(X) + R_K(X)\ F_{i+1}(X) \\ &= R_K(X)\ [F_i(X)\ S_i(X) + F_{i+1}(X)] \\ &= R_K(X)\ Q(X). \end{aligned}$$

Therefore, $R_K(X)$ divides $R_{i-1}(X)$. Since the sequence $R_0(X), R_1(X), \ldots, R_K(X), R_{K+1}(X)$ is finite, then by induction $R_K(X)$ divides both $R_0(X)$ and $R_1(X)$. Reversing the process, if $L(X)$ divides both $R_0(X)$ and $R_1(X)$, then

$$R_0(X) = L(X)\ W(X)$$

$$R_1(X) = L(X)\ Y(X)$$

for some polynomials $W(X)$ and $Y(X)$. But by the division algorithm

$$R_0(X) = R_1(X)\ S(X) + R_2(X).$$

Substitution yields

$$L(X)\ W(X) - L(X)\ Y(X)\ S(X) = R_2(X)$$

$$L(X)\ [W(X) - Y(X)\ S(X)] = R_2(X)$$

$$L(X)\ Z(X) = R_2(X).$$

Hence, $L(X)$ divides $R_2(X)$. Similarly if $L(X)$ divides $R_i(X)$ and $R_{i+1}(X)$, then $L(X)$ divides $R_{i+2}(X)$. Continuing inductively leads to the conclusion that $L(X)$ divides $R_K(X)$. Therefore, $R_K(X)$ divides $R_0(X)$ and $R_1(X)$ and any polynomial that divides $R_0(X)$ and $R_1(X)$ also divides $R_K(X)$. Thus, $R_K(X)$ is the greatest common divisor of $R_0(X)$ and $R_1(X)$.

The above theorem tells how to obtain the greatest common divisor of two polynomials. A machine orientated method is now developed for computing the sequence of $R_j(X)$'s. Beginning the sequence with $R_0(X)$ and $R_1(X)$, the polynomial $R_{i+1}(X)$ of the sequence is derived from $R_i(X)$ and $R_{i-1}(X)$ as follows: Let $R_{i-1}(X)$ of degree N_{i-1} be given by

$$R_{i-1}(X)$$
$$= r_{i-1,1}\ X^{N_{i-1}} + r_{i-1,2}\ X^{N_{i-1}-1} + \ldots + r_{i-1,N_{i-1}}\ X + r_{i-1,N_{i-1}+1}$$

and $R_i(X)$ of degree N_i be given by

$$R_i(X) = r_{i,1}\ X^{N_i} + r_{i,2}\ X^{N_i-1} + \ldots + r_{i,N_i}\ X + r_{i,N_i+1}$$

where $N_i \leq N_{i-1}$. Define $U_1(X)$ by

$$U_1(X) = (r_{i-1,1} / r_{i,1})\ X^{N_{i-1}-N_i}\ .$$

Then define $T_1(X)$ by $T_1(X) = R_{i-1}(X) - R_i(X)\ U_1(X)$ which can be expressed as

$$\begin{aligned} T_1(X) = {} & [r_{i-1,1} - r_{i,1}\ (r_{i-1,1} / r_{i,1})]\ X^{N_{i-1}} \\ & + [r_{i-1,2} - r_{i,2}\ (r_{i-1,1} / r_{i,1})]\ X^{N_{i-1}-1} \\ & + \dots \\ & + [r_{i-1,N_{i-1}+1} - r_{i,N_{i-1}+1}\ (r_{i-1,1} / r_{i,1})] \end{aligned}$$

where $r_{i,j} = 0$ for $j > N_i+1$, or by

$$T_1(X) = t_{1,1}\ X^{M_1} + t_{1,2}\ X^{M_1-1} + \dots + t_{1,M_1}\ X + t_{1,M_1+1}$$

where deg. $T_1(X) = M_1$. Three cases must now be considered.

(1) If $T_1(X) = 0$, then $R_i(X) = R_K(X)$; that is, $R_i(X)$ is the g.c.d. of $R_0(X)$ and $R_1(X)$.

(2) If $T_1(X) \neq 0$ and deg. $T_1(X) < N_i$, then $R_{i+1}(X) = T_1(X)$.

(3) If $T_1(X) \neq 0$ and deg. $T_1(X) \geq N_i$, then define $U_2(X)$ by

$$U_2(X) = (t_{1,1} / r_{i,1})\ X^{M_1-N_i}.$$

Define $T_2(X) = T_1(X) - U_2(X)\ R_i(X)$ which can be expressed by

$$\begin{aligned} T_2(X) = {} & [t_{1,1} - (t_{1,1} / r_{i,1})\ r_{i,1}]\ X^{M_1-1} \\ & + [t_{1,2} - (t_{1,1} / r_{i,1})\ r_{i,2}]\ X^{M_1-2} \end{aligned}$$

$$+ \ldots$$

$$+ [t_{1,M_1+1} - (t_{1,1}/r_{i,1})\, r_{i,M_1+1}]$$

where $r_{i,j} = 0$ for $j > N_i+1$. Again three cases are considered.

(1) If $T_2(X) = 0$, then $R_i(X)$ is the g.c.d. of $R_0(X)$ and $R_1(X)$.

(2) If $T_2(X) \neq 0$ and deg. $T_2(X) <$ deg. $R_i(X)$, then $R_{i+1}(X) = T_2(X)$.

(3) If $T_2(X) \neq 0$ and deg. $T_2(X) \geq N_i$, then similarly define $U_3(X)$ and $T_3(X)$ and repeat as above.

Since deg. $T_{i+1}(X) <$ deg. $T_i(X)$, then this process is finite (not to exceed N_{i-1}) ending, for some integer S, in $T_S(X)$ such that

(1) $T_S(X) = 0$ and $R_i(X)$ is the g.c.d. of $R_0(X)$ and $R_1(X)$ or

(2) $T_S(X) \neq 0$ but deg. $T_S(X) <$ deg. $R_i(X)$, in which case $T_S(X) = R_{i+1}(X)$.

Thus, using this algorithm and given $R_0(X)$ and $R_1(X)$, the sequence $R_0(X), R_1(X), R_2(X), \ldots, R_i(X), R_{i+1}(X)$ can be generated such that either

(1) $R_{i+1}(X) = 0$ and $R_i(X)$ is the g.c.d. of $R_0(X)$ and $R_1(X)$ or

(2) $R_{i+1}(X) \neq 0$ and $N_{i+1} < N_i$. In a finite number of iterations, $R_K(X)$, the g.c.d. of $R_0(X)$ and $R_1(X)$, can be obtained.

Recall that the desire is to obtain the polynomial $H(X) = P(X)/D(X)$ where $D(X)$ is the g.c.d. of $P(X)$ and $P'(X)$. Thus, after obtaining $D(X)$

by the above algorithm, it is necessary to divide P(X) by D(X) obtaining H(X) all whose zeros are simple.

Once H(X) is obtained, an appropriate method such as Newton's method is applied to extract the zeros of H(X). This gives all the zeros of P(X).

As in Newton's or Muller's method, the zeros may be checked for accuracy and possibly improved by using them as initial approximations with the particular method applied to the full (undeflated) polynomial, P(X).

2. Determining Multiplicities

After all zeros of P(X) are found, the multiplicity of each is determined as follows: If P(X) has m distinct zeros, then P(X) can be written as

$$P(X) = a_1(X - \alpha_1)^{e_1} (X - \alpha_2)^{e_2} \dots (X - \alpha_m)^{e_m}, \quad (m \leq N)$$

where α_i is a zero of P(X) and e_i is the multiplicity of α_i $(i = 1,2,\dots,m)$. Consider the zero α_j. Dividing out the term $(X - \alpha_j)$ by deflating P(X) using synthetic division gives $P_1(X)$ of degree N-1 which can be written as

$$P_1(X) = (X - \alpha_1)^{e_1} (X - \alpha_2)^{e_2} \dots (X - \alpha_j)^{e_j - 1} \dots (X - \alpha_m)^{e_m}.$$

Evaluating $P_1(X)$ at α_j gives $P_1(\alpha_j) = 0$ if $e_j > 1$. Thus, α_j is a zero of multiplicity at least equal to two. Then $P_1(X)$ is deflated obtaining $P_2(X)$ and $P_2(\alpha_j)$ is found. If $P_2(\alpha_j) = 0$, α_j is of multiplicity at least three. This process is continued until a deflated polynomial $P_K(X)$ is found such that either deg. $P_K(X) = 0$ or $P_K(\alpha_j) \neq 0$. Thus, if the value of the original polynomial, $P(X)$, and the next L successively deflated polynomials is zero at the root α_j and $P_{L+1}(\alpha_j) \neq 0$, then α_j is of multiplicity L+1.

The iteration formulas for deflation by synthetic division are derived in [6, PP. 106-107] and are given in Theorem 2.3 in Chapter II, § 1.

3. Procedure for the G.C.D. Method

The basic steps performed by the greatest common divisor method are listed sequentially as follows:

1. Given a polynomial, $P(X)$, in the form $P(X) = a_1X^N + a_2X^{N-1} + \ldots + a_NX + a_{N+1}$.
2. Calculate the derivative, $P'(X)$, of $P(X)$ in the form $P'(X) = b_1X^{N-1} + b_2X^{N-2} + \ldots + b_N$ where $b_1 = Na_1$, $b_2 = (N-1)a_2, \ldots, b_N = a_N$.
3. Find $D(X)$, the g.c.d. of $P(X)$ and $P'(X)$ using the algorithms developed above.
4. Calculate $H(X) = P(X)/D(X)$, the polynomial having only simple zeros.
5. Use some appropriate method to extract the zeros of $H(X)$.
6. Determine the multiplicity of each of the zeros obtained in step 5.

CHAPTER V

LEHMER'S METHOD

1. Derivation of the Algorithm

Lehmer's method in [4] is an iterative procedure designed to find numerical approximations to the zeros of polynomials with real or complex coefficients. The method employs a systematic search of the complex plane for the set of zeros which may be any arbitrary finite set of points. Lehmer's method has a slower convergence rate than methods with quadratic convergence but is well suited for high speed computers and is particularly useful as a subroutine by which polynomials created internally during the computer's work on a larger problem can be solved without external supervision since no initial approximation of a zero is required.

The heart of this search procedure lies in repeatedly applying and answering the basic question,

"Does a given polynomial have a zero inside a given circle?"

More precisely, the method can be divided into steps as follows. Let

$$P(X) = a_1X^N + a_2X^{N-1} + \ldots + a_NX + a_{N+1} \tag{5-1}$$

be a polynomial with complex coefficients such that $a_1 \neq 0$ and $P(0) \neq 0$.

Step 1. This step begins by asking the question, "Does P(X) have a zero inside the unit circle?" If the answer is yes, then the radius is repeatedly halved and the basic question asked until an annulus

$$R < |Z| < 2R \quad \text{(R is radius of inner circle)}$$

is found containing a zero, X_j, of $P(X)$ while the inner circle is free of zeros. Note that $R = \frac{1}{2^K}$ for some integer K since the radius is repeatedly halved. If the answer to the question is no, the radius is repeatedly doubled until the above conditions are satisfied where, in this case, $R = 2^M$ for some integer M. This resulting annulus can be completely covered by 8 overlapping circles (Figure 2.1) each of radius $r_1 = (\frac{5}{6})R$ with centers located at the vertices of a regular octagon inscribed in a circle of radius $(\frac{5}{3})R$, that is, the points

$$(\frac{5}{3})R\, e^{\frac{\Pi K}{4} i}, \quad K = 0,1,\ldots,7.$$

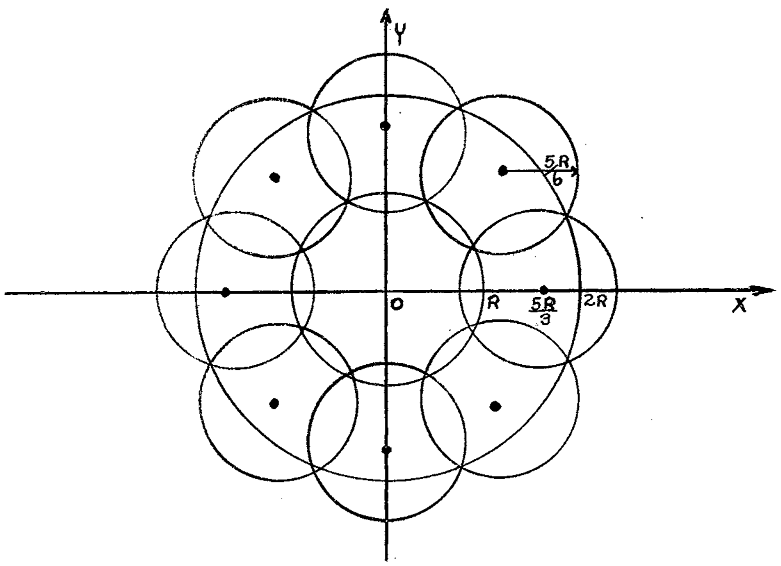


Figure 2.1. Covering an Annulus With Eight Overlapping Circles

Using the conversion $r\,e^{i\theta} = r\,(\cos\theta + i \sin\theta)$, the coordinates of these centers can be written in rectangular form as

$$(\tfrac{5}{3})R \cos(\tfrac{\Pi K}{4}) + i\,(\tfrac{5}{3})\,R \sin(\tfrac{\Pi K}{4}), \quad K = 0,1,\ldots,7.$$

Starting with K = 0, the basic question is asked of each of these circles until one is found containing the root X_j. Denoting the center of this circle by C_1 completes step 1.

Step 2. Using C_1 as the origin, step 2 repeatedly halves the radius, R, until an annulus

$$R_1 < |Z - C_1| < 2R_1$$

is found containing a zero while the inner circle of radius R_1 is free of roots. Here

$$R_1 = (\tfrac{1}{2^K})r_1 = \tfrac{1}{2^K}\,(\tfrac{5R}{6})$$

where K is a positive integer. $|Z - C_1|$ is the distance of a point Z in the annulus from the center C_1. Again this annulus can be covered by 8 overlapping circles each of radius $r_2 = (\tfrac{5}{6})R_1$ with centers at the points

$$\frac{5R_1}{3}\, e^{\frac{\Pi K}{4} i}, \quad K = 0,1,\ldots,7.$$

Note that r_1, r_2, R_1, and R are related by the following:

$$r_2 = \frac{5R_1}{6} = \frac{5}{6}\,(\frac{1}{2^K})r_1 \leq \frac{5}{6}\,(\frac{r_1}{2}) = \frac{5r_1}{12} = \frac{5}{12}\,(\frac{5R}{6}) = \frac{25R}{72}. \quad (5\text{-}1.1)$$

Beginning with K = 0, repeated application of the basic question to each of these 8 circles determines one containing X_j. Due to overlapping, it is possible for two circles to contain X_j, in which case, the circle corresponding to the smaller value of K is used. Denoting the center of this circle by C_2 completes step 2.

After the completion of step M of this procedure, a circle with center C_M and radius r_M is found containing the zero X_j. From repeated application of the inequality (5-1.1), we obtain

$$r_M \leq 2(\frac{5}{12})^M R.$$

Thus, by choosing M sufficiently large, a circle of sufficiently small radius is found containing the zero X_j. The coordinates of C_M with respect to the original origin, (0,0), of the complex plane is then the desired approximation to the zero, X_j.

After the zero X_j has been determined within the required degree of accuracy, the multiplicity of X_j is determined as explained in Chapter II, § 5. P(X) is then deflated by synthetic division according to Theorem 2.3 of Chapter II. This deflation removes X_j, as a point, from the complex plane, thus rendering X_j no longer available to be recaptured on any succeeding application of the method in extracting the remaining zeros. P(X) is then replaced by $P_1(X)$ and the above procedure applied to extract another root. Therefore, by successive deflations, all the zeros of P(X) can be approximated.

Figure 3 gives the geometric interpretation of Lehmer's method through step 3 of the above procedure.

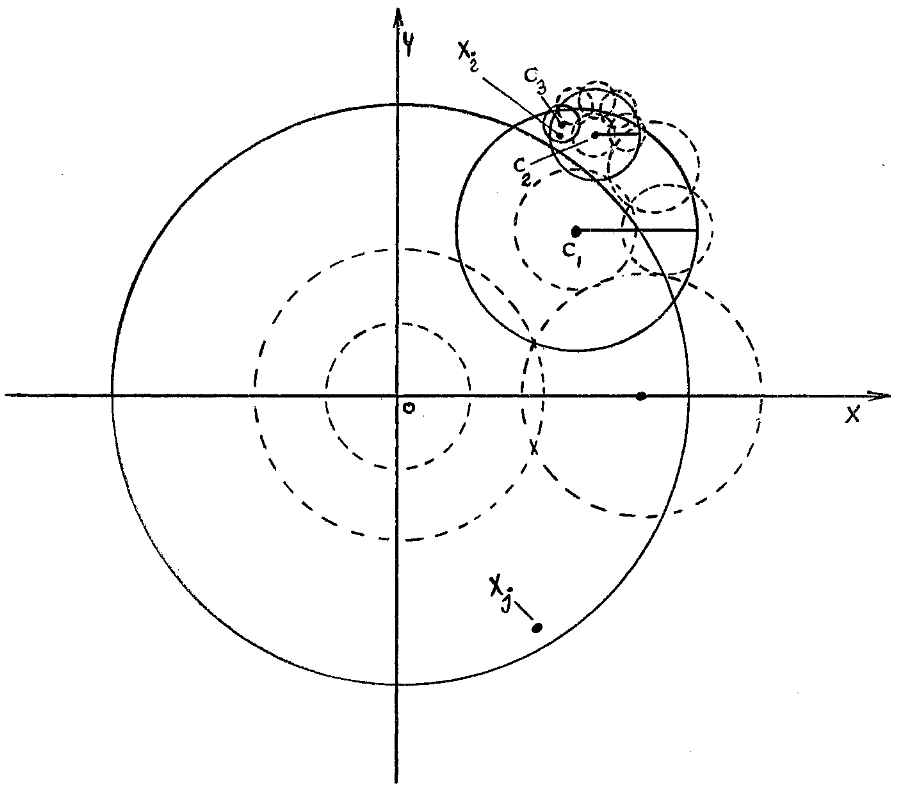


Figure 3. Geometric Interpretation of Lehmer's Method

The dotted circles indicate the absence of zeros within them. C_1,C_2, and C_3 are the centers of the resulting circles after completion of steps 1, 2, and 3, respectively. The zero determining the first annulus is X_j but X_i is captured by the process rather than X_j. Thus, if two zeros have approximately the same modulus, the one with the smaller argument is captured first. Lehmer's method extracts zeros roughly in the order of increasing modulus.

In order to apply the above procedure, a machine orientated method is needed to answer the basic question:

"Does $P(X)$ have a zero inside the circle with center C and radius ρ"; that is, the circle $|X - C| = \rho$?

By applying the transformation $Z = \rho Z$ which corresponds to a stretching of the complex plane if $\rho > 1$ or a shrinking of the complex plane if $\rho < 1$, and the translation $Z = Z + C$, the given circle, $|X - C| = \rho$ can be replaced by the unit circle, $|Z| = 1$, using the polynomial

$$G(Z) = P(\rho Z + C).$$

If X_j is a zero of $P(X)$ then $\beta = (X_j - C)/\rho$ is a zero of $G(X)$ since

$$\begin{aligned} G(\beta) &= P[\rho(\frac{X_j - C}{\rho}) + C] \\ &= P(X_j) \\ &= 0. \end{aligned}$$

Also $|\beta| < 1$ if and only if $|X_j - C| < \rho$; that is, β is a zero of $G(X)$ inside the unit circle if and only if X_j is a zero of $P(X)$ inside the circle $|X - C| = \rho$. Thus, the existence of the zero β implies the existence of the zero X_j and conversely. Thus, the basic question is equivalent to:

"Does $G(X)$ have a zero inside the unit circle Γ?"

In order to answer this question, a sequence of polynomials is constructed from $G(X)$ as follows: Let $G(X)$ be given by

$$G(X) = b_1X^N + b_2X^{N-1} + \dots + b_NX + b_{N+1} \qquad (5\text{-}2)$$

where b_1 and $b_{N+1} \neq 0$ and the coefficients are complex. Note that $G(X)$

is constructed from P(X) by the linear transformation $X = \rho X + c$ and hence deg. $G(X) = N =$ deg. $P(X)$. Denoting the conjugate of $b_j = u + iv$ by $\overline{b}_j = u - iv$, let $G^*(X)$ be defined by

$$G^*(X) = \overline{b}_{N+1}X^N + \overline{b}_N X^{N-1} + \ldots + \overline{b}_2 X + \overline{b}_1;$$

that is, the coefficients of $G^*(X)$ are the conjugates of those of $G(X)$ but in reverse order. Define the transformation T on G(X) by

$$T(G(X)) = \overline{b}_{N+1}G(X) = b_1 G^*(X).$$

T(G(X)) is a polynomial of degree less than deg. G(X) since

$$\begin{aligned} T(G(X)) &= \overline{b}_{N+1}(b_1 X^N + b_2 X^{N-1} + \ldots + b_N X + b_{N+1}) \\ &\quad - b_1(\overline{b}_{N+1}X^N + \overline{b}_N X^{N-1} + \ldots + \overline{b}_2 X + \overline{b}_1) \\ &= \overline{b}_{N+1}\, b_1 X^N + \overline{b}_{N+1} b_2 X^{N-1} + \ldots + \overline{b}_{N+1}\, b_N X + \overline{b}_{N+1}\, b_{N+1} \\ &\quad - b_1\, \overline{b}_{N+1}\, X^N - b_1\, \overline{b}_N X^{N-1} - \ldots - b_1\, \overline{b}_2 X - b_1\, \overline{b}_1 \\ &= (\overline{b}_{N+1} b_1 - b_1 \overline{b}_{N+1})X^N + (\overline{b}_{N+1} b_2 - b_1 \overline{b}_N)X^{N-1} \\ &\quad + \ldots + (\overline{b}_{N+1} b_N - b_1 \overline{b}_2)X + (\overline{b}_{N+1} b_{N+1} - b_1 \overline{b}_1) \\ &= t_1 X^{N-1} + t_2 X^{N-2} + \ldots + t_{N-1}X + t_N \end{aligned}$$

where

$$t_i = \overline{b}_{N+1}\, b_{i+1} - b_1\, \overline{b}_{N-i+1}, \quad i = 1,2,\ldots,N.$$

The constant term of T(G(X)) is

$$t_N = \overline{b}_{N+1}\, b_{N+1} - b_1\, \overline{b}_1 = |b_{N+1}|^2 - |b_1|^2 \qquad (5\text{-}3)$$

and is, therefore, real because the product of a complex number and its conjugate is real. In fact,

$$\begin{aligned} T(G(0)) &= \overline{b}_{N+1}\, b_{N+1} - b_1\, \overline{b}_1 \\ &= |b_{N+1}|^2 - |b_1|^2. \end{aligned} \qquad (5\text{-}4)$$

If $T(G(0)) \neq 0$, applying this transformation, T, to T(G(X)) gives $T^2(G(X)) = T[T(G(X))]$ where deg. $T^2(G(X)) <$ deg. $T(G(X))$. Continuing, we obtain a sequence of polynomials

$$T(G(X)),\ T^2(G(X)),\ T^3(G(X)),\ \ldots,\ T^K(G(X)) \qquad (5\text{-}5)$$

where

$$T^i(G(X)) = T[T^{i-1}(G(X))], \quad i = 2,\ldots,K.$$

Let deg. $T^i(G(X)) = d_i$. Then $N > d_1 > d_2 > \ldots > d_K \geq 0$ since, from above, it was shown that T reduces the degree of the polynomial by at least one.

The polynomial $T^{i+1}(G(X))$ is obtained from $T^i(G(X))$ as follows. Let $T^i(G(X))$ be denoted by

$$T^i(G(X)) = r_1 X^{d_i} + r_2 X^{d_i - 1} + \ldots + r_{d_i} X + r_{d_i+1},\ r_{d_i+1} \neq 0.$$

Then

$$[T^i(G(X))]^* = \bar{r}_{d_i+1}X^{d_i} + \bar{r}_{d_i}X^{d_i-1} + \ldots + \bar{r}_2X + \bar{r}_1.$$

T applied to $T^i(G(X))$ gives

$$\begin{aligned} T^{i+1}(G(X)) &= T[T^i(G(X))] \\ &= \bar{r}_{d_i+1}\, T^i(G(X)) - r_1[T^i(G(X))]^* \\ &= \sum_{j=1}^{d_i} \left(\bar{r}_{d_i+1}\, r_{j+1} - r_1\, \bar{r}_{d_i-j+1}\right) X^{d_i-j}. \end{aligned}$$

Since deg. $P(X) = N$ is finite, then in a finite number of steps (at most N+1) a polynomial $T^K(G(X))$ is obtained such that $T^K(G(0)) = 0$. This is due to the following reasoning. $T^i(G(0))$ is the constant, real term of $T^i(G(X))$. If $T^i(G(0)) = 0$, we are done. If $T^i(G(0)) \neq 0$, then in at most N steps, we obtain a polynomial $T^{K-1}(G(X))$ of degree 0; that is, $T^{K-1}(G(X)) = c$ (constant). Then $T^K(G(X)) = \bar{c}c - c\bar{c} = 0$. Hence $T^K(G(0)) = 0$.

Finally, the question, "Does the polynomial G(X) have a zero inside the unit circle Γ?" can be answered by the following two theorems.

<u>Theorem 5.2</u>. Let G(X) have no roots on the unit circle Γ. Suppose $G(0) \neq 0$. If for some $h > 0$, $T^h(G(0)) < 0$, then G(X) has at least one zero inside Γ. If, instead, $T^i(G(0)) > 0$ for $1 \leq i < K$ and $T^{K-1}(G(X))$ is a constant, then no roots of G(X) lie inside Γ.

The proof of this theorem can be found in [4, PP. 154-156]. The Cauchy Integral Formula, Rouche's Theorem, and the Argument Principle from complex analysis are used in the proof of Theorem 5.2. Statements and proofs of these theorems can be found in [8, PP. 293-295], [9, P. 145], and [9, PP. 142-143], respectively.

Theorem 5.3. Theorem 5.2 remains true if we weaken its hypothesis by deleting its first sentence. Let $G(X)$ be a polynomial with $G(0) \neq 0$. If for some $h > 0$, $T^h(G(0)) < 0$, then $G(X)$ has at least one zero inside the unit circle Γ. If, instead, $T^i(G(0)) > 0$ for $1 \leq i < K$ and $T^{K-1}(G(X))$ is a constant, then $G(X)$ has no roots inside Γ.

This theorem is proved in [4, PP. 156-157].

It may accidentally happen in certain specially made polynomials with integer coefficients that $T^{K-1}(G(X))$ of Theorem 5.2 is not a constant. For example, consider the polynomial

$$G(X) = 6X^4 - 35X^3 + 62X^2 - 35X + 6.$$

In this case $G^*(X) = G(X)$ so that $T(G(X))$ vanishes identically. When this happens, it is better to ask the basic question on a concentric circle of radius larger than the present one by a factor such as $\frac{3}{2}$. This also is good practice in case the value $T^i(G(0))$ is so small in magnitude that the sign is uncertain due to roundoff error, or when $G(0) = 0$.

To illustrate this algorithm, consider the following example. Let $G(X)$ be the polynomial

$$G(X) = (1 + i)X^2 + 2X + (2 + 2i).$$

Then

$$G^*(X) = (2 - 2i)X^2 + 2X + (1 - i)$$

$$\begin{aligned}
T(G(X)) &= (2 - 2i)\ G(X) - (1 + i)\ G^*(X) \\
&= (2 - 2i)[(1 + i)X^2 + 2X + 2 + 2i] \\
&\quad - (1 + i)[(2 - 2i)X^2 + 2X + (1 - i)] \\
&= [(2 - 2i)(1 + i) - (1 + i)(2 - 2i)]X^2 \\
&\quad + [(2 - 2i)2 - (1 + i)2]X \\
&\quad + [(2 - 2i)(2 + 2i) - (1 + i)(1 - i)] \\
&= (2 - 6i)X + 6
\end{aligned}$$

Thus, $T(G(0)) = 6$

$$T^*(G(X)) = 6X + (2 + 6i)$$

$$\begin{aligned}
T^2(G(X)) &= T[T(G(X))] \\
&= 6T(G(X)) - (2 - 6i)\ T^*(G(X)) \\
&= 6[(2 - 6i)X + 6] - (2 - 6i)[6X + (2 + 6i)] \\
&= [6(2 - 6i) - (2 - 6i)6]X + 36 - (2 - 6i)(2 + 6i) \\
&= 36 - (4 + 36) \\
&= 36 - 40 \\
&= -4
\end{aligned}$$

Thus, $T^2(G(0)) = -4$.

Therefore, by Theorem 5.3, $G(X)$ has at least one root inside Γ.

2. Convergence of Lehmer's Method

Convergence of this searching technique is somewhat slower than other methods but is somewhat that of the geometric progression of ratio $\frac{5}{12}$. It has been shown that after completion of step K, a circle of radius less than $2R(\frac{5}{12})^K$ is obtained containing a root. Twenty-seven steps give a radius of length less than $R \cdot 10^{-10}$. Hence, for K large enough, the center of the circle is a good approximation to the zero. If the value of P(X) at the center is sufficiently near zero, then convergence is obtained.

3. Determining Multiplicities

Multiplicities of the zeros are determined as described in Chapter II, § 5.

4. Procedure for Lehmer's Method

The basic steps performed by Lehmer's method are listed sequentially as follows:

1. Use Theorem 5.3 to determine if P(X) has a zero inside the unit circle with center at the origin.
    a. If the answer is yes, halve the radius until a circle of radius R is found such that P(X) has a zero inside the circle $|X| = 2R$ but none inside $|X| = R$.
    b. If the answer is no, double the radius until such an R is found.

2. Beginning with $K = 0$, translate the origin to the point $c = (\frac{5R}{3})(\text{Cos}(\frac{\Pi K}{4}) + i\ \text{Sin}(\frac{\Pi K}{4}))$ using the radius $\rho = (\frac{5}{6})R$. Call the resulting polynomial $G(X)$.

3. Using Theorem 5.3 determine if $G(X)$ has a zero inside this circle.

   a. If it does, go to step 4.

   b. If it does not, increase K by 1 and go to step 2 with R replaced by ρ.

   c. If Theorem 5.3 cannot be applied, replace R by $(\frac{3}{2})R$ and return to step 2.

4. Save this center as the next approximation to the zero and halve the radius until a circle of radius R_1 is found such that $G(X)$ has a zero inside $|X - c| = 2R_1$ but none in $|X - c| = R$.

5. Test for convergence. Is $P(c)$ sufficiently near zero to be called a root? If it is not, return to step 2. If convergence is obtained, then

   a. Save latest center as the approximation to the zero.

   b. Deflate $P(X)$ using $(X - c)$, resulting in $P_1(X)$.

   c. Replace $P(X)$ by $P_1(X)$.

   d. Return to step 1.

Given the polynomial $F(X) = f_1X^M + f_2X^{M-1} + \ldots + f_MX + f_{M+1}$, the sequence of steps to determine if $F(X)$ has a zero inside a circle of radius ρ and center c is listed in the order performed below.

Theorem 5.3 consists of steps 2 - 8.

1. Calculate $G(X) = F(\rho X + c)$ where ρ is the radius of the circle and c is its center.
   $G(X) = g_1X^S + g_2X^{S-1} + \ldots + g_SX + g_{S+1}$. This replaces the given circle with the unit circle Γ. Theorem 5.3 is applied to this $G(X)$. Set $j = 1$.

2. If $G(0) = 0$, replace ρ by $(\frac{3}{2})\rho$ and return to step 1.
   If $G(0) \neq 0$, proceed to step 3.

3. Calculate $G^*(X) = \bar{g}_{S+1}X^S + \bar{g}_SX^{S-1} + \ldots + \bar{g}_2X + \bar{g}_1$.

4. Calculate $T^j(G(X)) = \bar{g}_{S+1}\, G(X) - g_1\, G^*(X)$.

5. Find $T^j(G(0))$.
   a. If $T^j(G(0))$ is too small to determine sign, replace ρ by $(\frac{3}{2})\rho$ and return to step 1.
   b. If $T^j(G(0)) < 0$, go to step 6.
   c. If $T^j(G(0)) = 0$, go to step 7.
   d. If $T^j(G(0)) > 0$, go to step 8.

6. $G(X)$ has a root inside Γ.

7. If $T^{j-1}(G(X))$ is a constant, $G(X)$ has no root inside Γ.
   If $T^{j-1}(G(X))$ is not a constant, replace ρ by $(\frac{3}{2})\rho$ and return to step 1.

8. Replace $G(X)$ by $T^j(G(X))$, increase j by 1 and return to step 2.

5. Conclusion of the Method

In applying the transformation T, the coefficients of $T^K(G(X))$ may become very large or very small in absolute value. For example, the constant term of T(G(X)) is computed by equation (5-3). If the magnitude of the constant term of G(X) is large compared to that of the leading coefficient, then repeated application of the transformation results in the constant term of $T^{i+1}(G(X))$ of sequence (5-5) having magnitude approximately double that of the constant term of $T^i(G(X))$. This problem was encountered when the program was run on the IBM S/360 which permits magnitudes between 10^{-77} and 10^{77}. The program could not be run on the IBM S/360 due to the limitation on the size of the exponent permitted.

CHAPTER VI

QUOTIENT-DIFFERENCE METHOD

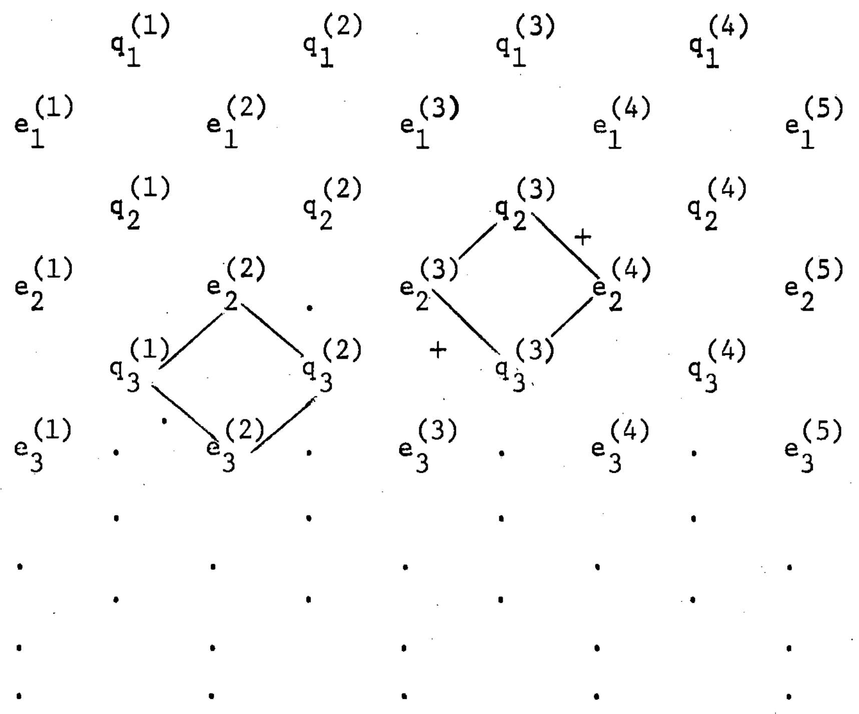
1. Derivation of the Algorithm

The quotient-difference (Q-D) method in [3, PP. 162-179] is an iterative procedure producing simultaneous approximations to all zeros of a polynomial with real or complex coefficients. This method has the particular advantage that no initial approximations to the zeros are required, since no information is needed other than the degree of the polynomial and its coefficients. Due to roundoff error and slow convergence, the Q-D method is recommended only for the purpose of giving initial approximations to the zeros. These initial approximations are then used with another method, such as Newton's method, for obtaining final results.

Let P(X) be a polynomial of degree N given by

$$P(X) = a_1X^N + a_2X^{N-1} + \dots + a_NX + a_{N+1} \qquad (6\text{-}1)$$

where $a_1 \neq 0 \neq a_{N+1}$ and the coefficients are complex numbers. The quotient-difference algorithm consists of constructing, row after row, a two dimensional array of numbers called the Q-D scheme arranged as indicated below for the case of $N = 4$. The notation here is chosen to eliminate zero subscripts and does not agree with the notation in [3], [10], or [11].



$q_1^{(1)}$ $q_1^{(2)}$ $q_1^{(3)}$ $q_1^{(4)}$
$e_1^{(1)}$ $e_1^{(2)}$ $e_1^{(3)}$ $e_1^{(4)}$ $e_1^{(5)}$
$q_2^{(1)}$ $q_2^{(2)}$ $q_2^{(3)}$ $q_2^{(4)}$
+
$e_2^{(1)}$ $e_2^{(2)}$ $e_2^{(3)}$ $e_2^{(4)}$ $e_2^{(5)}$
$q_3^{(1)}$ $q_3^{(2)}$ + $q_3^{(3)}$ $q_3^{(4)}$
$e_3^{(1)}$ $e_3^{(2)}$ $e_3^{(3)}$ $e_3^{(4)}$ $e_3^{(5)}$

Figure 3.1. The Q-D Scheme

The entries in the array are related by the rhombus rules:

$$q_{n+1}^{(K)} = (e_n^{(K+1)} - e_n^{(K)}) + q_n^{(K)} \qquad (K = 1,2,\ldots,N) \tag{6-2}$$

$$e_{n+1}^{(K)} = \frac{q_{n+1}^{(K)}}{q_{n+1}^{(K-1)}} \, e_n^{(K)} \qquad (K = 2,3,\ldots,N+1).$$

where n = 1,2,3,... . As shown in the above diagram, these rules can be remembered as follows:

1. In a rhombus configuration with an e-element on top, the product of the northeast pair is equal to the product of the southwest pair.

2. In a rhombus configuration with a q-element on top, the sum of the northeast pair is equal to the sum of the southwest pair.

It is helpful to observe the following facts concerning the above diagram:

1. There are 2N+1 columns.
2. The superscripts in each column are constant.
3. The subscripts in each row are constant.

Since each of the rhombus rules involves four adjacent elements, namely the vertices of a rhombus in the array, then the first two rows, the first column, and the last column must be known before the rules can be applied to generate, row after row, the remainder of the array. The first two rows are determined from the coefficients of the polynomial as follows:

$$q_1^{(1)} = -a_2/a_1$$

$$q_1^{(K)} = 0 \quad (K = 2,3,\ldots,N)$$

$$e_1^{(K)} = a_{K+1}/a_K \quad (K = 2,3,\ldots,N).$$

Here it is assumed that the coefficients are non-zero. The case in which some coefficients are zero will be treated later. The first and last column consists entirely of zeros:

$$e_n^{(1)} = e_n^{(N+1)} = 0$$

The array now appears as follows for

$$P(X) = a_1X^4 + a_2X^3 + a_3X^2 + a_4X + a_5.$$

$$\begin{array}{ccccccccc}
\underline{e_n^{(1)}} & \underline{q_n^{(1)}} & \underline{e_n^{(2)}} & \underline{q_n^{(2)}} & \underline{e_n^{(3)}} & \underline{q_n^{(3)}} & \underline{e_n^{(4)}} & \underline{q_n^{(4)}} & \underline{e_n^{(5)}} \\
 & -\frac{a_2}{a_1} & & 0 & & 0 & & 0 & \\
0 & & \frac{a_3}{a_2} & & \frac{a_4}{a_3} & & \frac{a_5}{a_4} & & 0 \\
 & q_2^{(1)} & & q_2^{(2)} & & q_2^{(3)} & & q_2^{(4)} & \\
0 & & e_2^{(2)} & & e_2^{(3)} & & e_2^{(4)} & & 0 \\
 & q_3^{(1)} & & q_3^{(2)} & & q_3^{(3)} & & q_3^{(4)} & \\
0 & \cdot & e_3^{(2)} & \cdot & e_3^{(3)} & \cdot & e_3^{(4)} & \cdot & 0 \\
 & \cdot & & \cdot & & \cdot & & \cdot & \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
\cdot & & \cdot & & \cdot & & \cdot & & \cdot \\
\cdot & & \cdot & & \cdot & & \cdot & & \cdot
\end{array}$$

Figure 3.2. Starting the Q-D Scheme

The rhombus rules can be obtained as in [11, PP. 36-38]. This is accomplished as follows, keeping in mind that the notation is different from that in [11]. Let P(X) be a polynomial of degree N given by equation (6-1). Assume for the present that the zeros $X_1, X_2, \ldots, X_N$

are all simple and satisfy $|X_1| > |X_2| > \dots > |X_N| > 0$. Assume further that the coefficients are non-zero. P(X) can be expressed in the form

$$P(X) = a_1 \prod_{i=1}^{N} (X - X_i)$$

or

$$\frac{1}{P(X)} = \frac{1}{a_1 \prod_{i=1}^{N} (X - X_i)} .$$

Then $\frac{1}{P(X)}$ can be expressed as a sum of partial fractions:

$$\frac{1}{P(X)} = \frac{A_1}{X - X_1} + \frac{A_2}{X - X_2} + \dots + \frac{A_N}{X - X_N} . \tag{6-3}$$

To verify that this is true, it is only necessary to show that there exists unique $A_1, A_2, \dots, A_N$ such that equation (6-3) holds. This is done as follows. Combining terms in (6-3) results in

$$\frac{1}{P(X)} = \frac{A_1 \prod_{i=2}^{N} (X - X_i) + \dots + A_j \prod_{\substack{i=1 \\ i \neq j}}^{N} (X - X_i) + \dots + A_N \prod_{i=1}^{N-1} (X - X_i)}{\prod_{i=1}^{N} (X - X_i)} . \tag{6-4}$$

Substituting into the left hand side of equation (6-4) gives

$$\frac{1}{a_1 \prod_{i=1}^{N} (X - X_i)} = \frac{\sum_{j=1}^{N} \left(A_j \prod_{\substack{i=1 \\ i \neq j}}^{N} (X - X_i) \right)}{\prod_{i=1}^{N} (X - X_i)} .$$

$$\frac{1}{a_1} = \sum_{j=1}^{N} \left(A_j \prod_{\substack{i=1 \\ i \neq j}}^{N} (X - X_i) \right) .$$

Consider $\frac{1}{a_1}$ as a polynomial of degree N-1 all of whose coefficients are zero except the constant coefficient $\frac{1}{a_1}$. The right hand side is also a polynomial of degree N-1 since each of the N terms is the product of N-1 linear factors. Equating coefficients of these two polynomials gives N simultaneous equations in the N unknowns $A_1, A_2, \ldots, A_N$. Solution of this system yields the result. The terms on the right hand side of equation (6-3) can be expanded in a geometric series.

$$\frac{A_r}{X - X_r} = \frac{A_r}{X} \left(\frac{1}{1 - \frac{X_r}{X}} \right)$$

$$= A_r \left(\frac{1}{X} + \frac{X_r}{X^2} + \frac{X_r^2}{X^3} + \ldots \right) .$$

for all X such that $X > X_r$, since the sum of the geometric series

$$\frac{1}{X} + \frac{X_r}{X^2} + \frac{X_r^2}{X^3} + \ldots$$

is

$$\frac{\frac{1}{X}}{1 - \frac{X_r}{X}} = \frac{1}{X}\left(\frac{1}{1 - \frac{X_r}{X}}\right), \quad \left|\frac{X_r}{X}\right| < 1.$$

Thus, if $X > X_1$, then $X > X_r$ for $r = 1,2,\ldots,N$, and

$$\begin{aligned}
\frac{1}{P(X)} = A_1 &\left(\frac{1}{X} + \frac{X_1}{X^2} + \frac{X_1^2}{X^3} + \ldots\right) \\
&+ A_2 \left(\frac{1}{X} + \frac{X_2}{X^2} + \frac{X_2^2}{X^3} + \ldots\right) \\
&+ \ldots \\
&+ A_N \left(\frac{1}{X} + \frac{X_N}{X^2} + \frac{X_N^2}{X^3} + \ldots\right) \\
&= \sum_{j=1}^{N} A_j \left(\frac{1}{X}\right) + \sum_{j=1}^{N} A_j \left(\frac{X_j}{X^2}\right) + \sum_{j=1}^{N} A_j \left(\frac{X_j^2}{X^3}\right) + \ldots \\
&= \sum_{i=0}^{\infty} \left[\sum_{j=1}^{N} A_j \left(\frac{X_j^i}{X^{i+1}}\right)\right] .
\end{aligned}$$

Let

$$\alpha_i = \sum_{j=1}^{N} A_j X_j^i .$$

Then

$$\frac{1}{P(X)} = \sum_{i=0}^{\infty} \alpha_i \frac{1}{X^{i+1}} .$$

Computing the quotient between two consecutive coefficients gives

$$q_i^{(1)} = \frac{\alpha_{i+1}}{\alpha_i} = \frac{A_1X_1^{i+1} + A_2X_2^{i+1} + A_3X_3^{i+1} + \ldots + A_NX_N^{i+1}}{A_1X_1^i + A_2X_2^i + A_3X_3^i + \ldots + A_NX_N^i} .$$

Multiplying both numerator and denominator by $\frac{1}{A_1X_1^{i+1}}$ gives

$$q_i^{(1)} = \frac{A_1X_1^{i+1}\left(\frac{1}{A_1X_1^{i+1}}\right) + A_2X_2^{i+1}\left(\frac{1}{A_1X_1^{i+1}}\right) + \ldots + A_NX_N^{i+1}\left(\frac{1}{A_1X_1^{i+1}}\right)}{A_1X_1^{i}\left(\frac{1}{A_1X_1^{i+1}}\right) + A_2X_2^{i}\left(\frac{1}{A_1X_1^{i+1}}\right) + \ldots + A_NX_N^{i}\left(\frac{1}{A_1X_1^{i+1}}\right)}$$

$$= \frac{1 + \frac{A_2}{A_1}\left(\frac{X_2}{X_1}\right)^{i+1} + \ldots + \frac{A_N}{A_1}\left(\frac{X_N}{X_1}\right)^{i+1}}{\frac{1}{X_1}\left[1 + \frac{A_2}{A_1}\left(\frac{X_2}{X_1}\right)^{i} + \ldots + \frac{A_N}{A_1}\left(\frac{X_N}{X_1}\right)^{i}\right]} .$$

But $|X_K/X_1| < 1$ for $K = 2,3,\ldots,N$. Thus, as $i \to \infty$ the numerator approaches 1 while the denominator approaches $\frac{1}{X_1}$. Thus, we have

$$\lim_{i\to\infty} q_i^{(1)} = \frac{1}{\left(\frac{1}{X_1}\right)} = X_1 .$$

Then

$$\lim_{i\to\infty} \frac{X_1 - q_i^{(1)}}{\left(\frac{X_2}{X_1}\right)^i} = \lim_{i\to\infty} X_1 \left[\frac{1 - \dfrac{1 + \frac{A_2}{A_1}\left(\frac{X_2}{X_1}\right)^{i+1} + \ldots + \frac{A_N}{A_1}\left(\frac{X_N}{X_1}\right)^{i+1}}{1 + \frac{A_2}{A_1}\left(\frac{X_2}{X_1}\right)^{i} + \ldots + \frac{A_N}{A_1}\left(\frac{X_N}{X_1}\right)^{i}}}{\left(\frac{X_2}{X_1}\right)^i}\right] .$$

By combining terms in the numerator, combining like terms, and factoring, the right hand side of the preceding equation becomes

$$\lim_{i\to\infty} X_1 \left[\frac{\dfrac{1 + \dfrac{A_2}{A_1}\left(\dfrac{X_2}{X_1}\right)^i + \ldots + \dfrac{A_N}{A_1}\left(\dfrac{X_N}{X_1}\right)^i - 1 - \dfrac{A_2}{A_1}\left(\dfrac{X_2}{X_1}\right)^{i+1} - \ldots - \dfrac{A_N}{A_1}\left(\dfrac{X_N}{X_1}\right)^{i+1}}{1 + \dfrac{A_2}{A_1}\left(\dfrac{X_2}{X_1}\right)^i + \ldots + \dfrac{A_N}{A_1}\left(\dfrac{X_N}{X_1}\right)^i}}{\left(\dfrac{X_2}{X_1}\right)^i} \right]$$

$$= \lim_{i\to\infty} X_1 \left[\frac{\dfrac{A_2}{A_1}\left(\dfrac{X_2}{X_1}\right)^i \left(1 - \dfrac{X_2}{X_1}\right) + \dfrac{A_3}{A_1}\left(\dfrac{X_3}{X_1}\right)^i \left(1 - \dfrac{X_3}{X_1}\right) + \ldots + \dfrac{A_N}{A_1}\left(\dfrac{X_N}{X_1}\right)^i \left(1 - \dfrac{X_N}{X_1}\right)}{\left(\dfrac{X_2}{X_1}\right)^i \left(1 + \dfrac{A_2}{A_1}\left(\dfrac{X_2}{X_1}\right)^i + \ldots + \dfrac{A_N}{A_1}\left(\dfrac{X_N}{X_1}\right)^i\right)} \right]$$

$$= \lim_{i\to\infty} X_1 \left[\frac{\dfrac{A_2}{A_1}\left(1 - \dfrac{X_2}{X_1}\right) + \dfrac{A_3}{A_1}\left(\dfrac{X_3}{X_2}\right)^i \left(1 - \dfrac{X_3}{X_1}\right) + \ldots + \dfrac{A_N}{A_1}\left(\dfrac{X_N}{X_2}\right)^i \left(1 - \dfrac{X_N}{X_1}\right)}{1 + \dfrac{A_2}{A_1}\left(\dfrac{X_2}{X_1}\right)^i + \ldots + \dfrac{A_N}{A_1}\left(\dfrac{X_N}{X_1}\right)^i} \right]$$

But $|X_K/X_2| < 1$ for $K = 3,4,\ldots,N$ and $|X_K/X_1| < 1$ for $K = 2,3,\ldots,N$. Thus

$$\lim_{i\to\infty} \frac{X_1 - q_i^{(1)}}{\left(\dfrac{X_2}{X_1}\right)^i} = X_1 \left[\frac{A_2}{A_1}\left(1 - \frac{X_2}{X_1}\right)\right]. \tag{6-5}$$

Similarly, replacing i by i+1:

$$\lim_{i\to\infty} \frac{X_1 - q_{i+1}^{(1)}}{\left(\frac{X_2}{X_1}\right)^i}$$

$$= \lim_{i\to\infty} X_1 \left[\frac{1 - \left(\frac{1 + \frac{A_2}{A_1}\left(\frac{X_2}{X_1}\right)^{i+2} + \ldots + \frac{A_N}{A_1}\left(\frac{X_N}{X_1}\right)^{i+2}}{1 + \frac{A_2}{A_1}\left(\frac{X_2}{X_1}\right)^{i+1} + \ldots + \frac{A_N}{A_1}\left(\frac{X_N}{X_1}\right)^{i+1}} \right)}{\left(\frac{X_2}{X_1}\right)^i} \right]$$

$$= \lim_{i\to\infty} X_1 \left[\frac{1 + \frac{A_2}{A_1}\left(\frac{X_2}{X_1}\right)^{i+1} + \ldots + \frac{A_N}{A_1}\left(\frac{X_N}{X_1}\right)^{i+1} - 1 - \frac{A_2}{A_1}\left(\frac{X_2}{X_1}\right)^{i+2} - \ldots - \frac{A_N}{A_1}\left(\frac{X_N}{X_1}\right)^{i+2}}{\left(\frac{X_2}{X_1}\right)^i \left(1 + \frac{A_2}{A_1}\left(\frac{X_2}{X_1}\right)^{i+1} + \ldots + \frac{A_N}{A_1}\left(\frac{X_N}{X_1}\right)^{i+1}\right)} \right]$$

$$= \lim_{i\to\infty} X_1 \left[\frac{\frac{A_2}{A_1}\left(\frac{X_2}{X_1}\right)^{i+1}\left(1 - \frac{X_2}{X_1}\right) + \frac{A_3}{A_1}\left(\frac{X_3}{X_1}\right)^{i+1}\left(1 - \frac{X_3}{X_1}\right) + \ldots + \frac{A_N}{A_1}\left(\frac{X_N}{X_1}\right)^{i+1}\left(1 - \frac{X_N}{X_1}\right)}{\left(\frac{X_2}{X_1}\right)^i \left(1 + \frac{A_2}{A_1}\left(\frac{X_2}{X_1}\right)^{i+1} + \ldots + \frac{A_N}{A_1}\left(\frac{X_N}{X_1}\right)^{i+1}\right)} \right]$$

$$= \lim_{i\to\infty} X_1 \left[\frac{\frac{A_2}{A_1}\left(\frac{X_2}{X_1}\right)\left(1 - \frac{X_2}{X_1}\right) + \frac{A_3}{A_1}\left(\frac{X_3}{X_2}\right)^{i+1}\left(1 - \frac{X_3}{X_1}\right) + \ldots + \frac{A_N}{A_1}\left(\frac{X_N}{X_2}\right)^{i+1}\left(1 - \frac{X_N}{X_1}\right)}{1 + \frac{A_2}{A_1}\left(\frac{X_2}{X_1}\right)^{i+1} + \ldots + \frac{A_N}{A_1}\left(\frac{X_N}{X_1}\right)^{i+1}} \right].$$

Thus

$$\lim_{i\to\infty} \frac{X_1 - q_{i+1}^{(1)}}{\left(\frac{X_2}{X_1}\right)^i} = X_1 \left(\frac{A_2}{A_1}\right)\left(\frac{X_2}{X_1}\right)\left(1 - \frac{X_2}{X_1}\right)$$

$$= X_2 \left(\frac{A_2}{A_1}\right)\left(1 - \frac{X_2}{X_1}\right). \qquad (6\text{-}6)$$

Subtracting equation (6-6) from equation (6-5) yields

$$\lim_{i\to\infty} \frac{q_{i+1}^{(1)} - q_i^{(1)}}{\left(\frac{X_2}{X_1}\right)^i} = (X_1 - X_2)\left(\frac{A_2}{A_1}\right)\left(1 - \frac{X_2}{X_1}\right).$$

Similarly, as in deriving equations (6-5) and (6-6),

$$\lim_{i\to\infty} \frac{X_1 - q_{i+2}^{(1)}}{\left(\frac{X_2}{X_1}\right)^i} = \frac{X_2^2}{X_1}\left(\frac{A_2}{A_1}\right)\left(1 - \frac{X_2}{X_1}\right).$$

Then

$$\lim_{i\to\infty} \frac{q_{i+2}^{(1)} - q_{i+1}^{(1)}}{\left(\frac{X_2}{X_1}\right)^i} = \left(X_2 - \frac{X_2^2}{X_1}\right)\frac{A_2}{A_1}\left(1 - \frac{X_2}{X_1}\right).$$

Now let

$$e_i^{(2)} = q_{i+1}^{(1)} - q_i^{(1)}.$$

Then

$$\lim_{i\to\infty} \frac{e_{i+1}^{(2)}}{e_i^{(2)}} = \lim_{i\to\infty} \frac{q_{i+2}^{(1)} - q_{i+1}^{(1)}}{q_{i+1}^{(1)} - q_i^{(1)}}$$

$$= \lim_{i\to\infty} \frac{\dfrac{q_{i+2}^{(1)} - q_{i+1}^{(1)}}{\left(\dfrac{X_2}{X_1}\right)^i}}{\dfrac{q_{i+1}^{(1)} - q_i^{(1)}}{\left(\dfrac{X_2}{X_1}\right)^i}}$$

$$= \frac{X_2 - \dfrac{X_2^2}{X_1}}{X_1 - X_2}$$

$$= \frac{(X_1 - X_2)\left(\dfrac{X_2}{X_1}\right)}{(X_1 - X_2)}$$

$$= \frac{X_2}{X_1} \; .$$

Eliminating X_1 we have

$$\left(\lim_{i\to\infty} \frac{e_{i+1}^{(2)}}{e_i^{(2)}}\right) X_1 = X_2$$

$$\left(\lim_{i\to\infty} \frac{e_{i+1}^{(2)}}{e_i^{(2)}}\right)\left(\lim_{i\to\infty} q_i^{(1)}\right) = X_2 .$$

But

$$\lim_{i\to\infty} q_i^{(1)} = \lim_{i\to\infty} q_{i+1}^{(1)} \; .$$

Hence

$$\underset{i\to\infty}{\text{limit}} \frac{e_{i+1}^{(2)}}{e_i^{(2)}} q_{i+1}^{(1)} = X_2.$$

Further, let

$$q_{i+1}^{(2)} = \frac{e_{i+1}^{(2)}}{e_i^{(2)}} q_{i+1}^{(1)} .$$

Then we have the following:

$$\underset{i\to\infty}{\text{limit}}\ q_i^{(1)} = X_1$$

$$\underset{i\to\infty}{\text{limit}}\ q_{i+1}^{(2)} = \underset{i\to\infty}{\text{limit}}\ q_i^{(2)} = X_2.$$

Let

$$e_i^{(3)} = q_{i+1}^{(2)} - q_i^{(2)} + e_i^{(2)}.$$

Then as before

$$\underset{i\to\infty}{\text{limit}} \frac{e_{i+1}^{(3)}}{e_i^{(3)}} = \frac{X_3}{X_2}$$

$$\underset{i\to\infty}{\text{limit}} \frac{e_{i+1}^{(3)}}{e_i^{(3)}} q_{i+1}^{(2)} = X_3$$

$$\frac{e_{i+1}^{(3)}}{e_i^{(3)}} q_{i+1}^{(2)} = q_{i+1}^{(3)}.$$

Thus

$$\lim_{i\to\infty} q_{i+1}^{(3)} = \lim_{i\to\infty} q_i^{(3)} = X_3.$$

Continuing by induction, the following results are established:

1. The rhombus rules have been obtained. They are

$$q_{n+1}^{(K)} = (e_n^{(K+1)} - e_n^{(K)}) + q_n^{(K)} \quad (K = 1,2,\ldots,N)$$

$$e_{n+1}^{(K)} = \frac{q_{n+1}^{(K)}}{q_{n+1}^{(K-1)}} e_n^{(K)} \quad (K = 2,3,\ldots,N+1)$$

where $n = 1,2,3,\ldots$.

2. $\lim_{n\to\infty} q_n^{(K)} = X_K.$

2. Existence of the Q-D Scheme

The Q-D scheme described previously may fail to exist if one or more of the divisors $q_{n+1}^{(K-1)}$ of the rhombus rules above is zero. It appears to be difficult to give explicit necessary and sufficient conditions for existence of the scheme in terms of the polynomial P(X). One sufficient condition in terms of the zeros of P(X), given in [3, P. 165], is that the zeros $X_1,X_2,\ldots,X_N$ are simple and have distinct absolute values such that

$$|X_1| > |X_2| > \ldots > |X_N| > 0.$$

3. Convergence Properties

Let the zeros $X_1,X_2,\ldots,X_N$ of P(X) be numbered in decreasing magnitude; that is,

$$|X_1| \geq |X_2| \geq \dots \geq |X_N|.$$

Then [10, P. 571] gives the following theorems.

<u>Theorem 6.1</u>. For every K such that $|X_{K-1}| > |X_K| > |X_{K+1}|$,

$$\underset{n\to\infty}{\text{limit}}\ q_n^{(K)} = X_K\ ;$$

that is, the Kth column of the array converges to the Kth zero of P(X).

The proof of this theorem has already been established in the derivation of the algorithm.

<u>Theorem 6.2</u>. For every K such that $|X_K| > |X_{K+1}|$

$$\underset{n\to\infty}{\text{limit}}\ e_n^{(K)} = 0.$$

Proof: From the rhombus rules for K = 1 we obtain

$$q_{n+1}^{(1)} = e_n^{(2)} - e_n^{(1)} + q_n^{(1)}$$

$$\underset{n\to\infty}{\text{limit}}\ e_n^{(2)} = \underset{n\to\infty}{\text{limit}}\ q_{n+1}^{(1)} - \underset{n\to\infty}{\text{limit}}\ q_n^{(1)} + \underset{n\to\infty}{\text{limit}}\ e_n^{(1)}.$$

But $e_n^{(1)} = 0$ for all n and $\underset{n\to\infty}{\text{limit}}\ q_n^{(K)} = X_K$ implies that

$$\underset{n\to\infty}{\text{limit}}\ e_n^{(2)} = X_1 - X_1 + 0 = 0.$$

By induction we conclude that $\underset{n\to\infty}{\text{limit}}\ e_n^{(K)} = 0$.

To this point, the existence of the scheme and convergence properties have been established for the case where the coefficients are non-zero and the zeros are simple and satisfy

$$|X_1| > |X_2| > \dots > |X_N| > 0.$$

The following observations should be noted:

1. The e-columns which approach zero (0) as a limit divide the Q-D scheme into subtables.
2. All zeros whose subscripts agree with the superscripts of the q-columns in a subtable have the same modulus.

Thus, if zero X_K is the only zero of its modulus, then

$$\lim_{n\to\infty} e_n^{(K)} = 0, \quad \lim_{n\to\infty} e_n^{(K+1)} = 0, \quad \lim_{n\to\infty} q_n^{(K)} = X_K.$$

4. Zeros of Equal Magnitude

Consider the case where M zeros, $X_{K+1}, X_{K+2}, \dots, X_{K+M}$, have the same modulus; that is,

$$|X_K| > |X_{K+1}| = |X_{K+2}| = \dots = |X_{K+M}| > |X_{K+M+1}|.$$

As noted above, the q-columns $q_n^{(K+1)}$, $q_n^{(K+2)}, \dots, q_n^{(K+M)}$ will be in the same subtable and the e-columns within that subtable, $e_n^{(K+2)}, \dots,$ $e_n^{(K+M)}$, will not tend to zero. The zeros $X_{K+1}, X_{K+2}, \dots, X_{K+M}$ are then obtained from the following theorem given in [3, P. 167]. A proof of this result can be found in [12, PP. 42-43].

Theorem 6.3. Suppose P(X) has M zeros of equal moduli satisfying

$$|X_K| > |X_{K+1}| = |X_{K+2}| = \dots = |X_{K+M}| > |X_{K+M+1}|.$$

Construct sequences of polynomials $P_n^{(j)}(X)$, $j = K, K+1, \dots, K+M$ by means of the following recurrance relations:

$$P_n^{(K)}(X) = 1, \quad n = 1,2,3,\dots$$

$$P_n^{(i)}(X) = X\, P_{n+1}^{(i-1)}(X) - q_n^{(i)}\, P_n^{(i-1)}(X) \qquad i = K+1,\dots,K+M$$
$$n = 1,2,\dots\ .$$

Then $\lim\limits_{n\to\infty} P_n^{(K+M)}(X) = (X - X_{K+1})(X - X_{K+2}) \dots (X - X_{K+M})$; that is, the coefficients of the polynomials $P_n^{(K+M)}(X)$ tend, as $n\to\infty$, to the coefficients of the polynomial with zeros $X_{K+1}, X_{K+2}, \dots, X_{K+M}$ and leading coefficient 1.

These polynomials can be thought of as being arranged in a two dimensional scheme, similar to the Q-D scheme as follows, where the $q_n^{(j)}$'s are obtained from the Q-D scheme.

The practical application of this theorem is as follows. Construct the Q-D scheme for n sufficiently large to determine convergence or non-convergence of the q-columns; that is, to divide the Q-D scheme into subtables. Convergence of a q-column corresponds to a zero of isolated absolute value. Divergence of M adjacent q-columns indicates that there are M zeros of equal moduli. Also the e-columns which converge to zero divide the scheme into subtables. Beginning with the first divergent q-column, one constructs the sequences of polynomials

$$P_n^{(K)}(X),\ P_n^{(K+1)}(X),\ \dots,\ P_n^{(K+M)}(X).$$

The roots $X_{K+1},\ldots,X_{K+M}$ are then found as the zeros of

$$\lim_{n\to\infty} P_n^{(K+M)}(X).$$

$$\begin{tikzcd}
1 & P_1^{(K+1)}(X) & P_1^{(K+2)}(X) & \cdots & P_1^{(K+M)}(X) \\
1 & P_2^{(K+1)}(X) & P_2^{(K+2)}(X) & \cdots & P_2^{(K+M)}(X) \\
\vdots & \vdots & \vdots & & \vdots \\
1 & P_n^{(K+1)}(X) \arrow[r, "-q_n^{(K+2)}"] & P_n^{(K+2)}(X) & \cdots & P_n^{(K+M)}(X) \\
1 & P_{n+1}^{(K+1)}(X) \arrow[ur, "X"] & \vdots & & \vdots \\
\vdots & \vdots & & &
\end{tikzcd}$$

Figure 3.3. Scheme for the Polynomials of Theorem 6.3

5. Zero Coefficients of the Polynomial P(X)

Finally consider the case in which some of the coefficients $a_2,a_3,\ldots,a_N$ of P(X) are zero. Let $X^* = X - a$, $a \neq 0$, for some suitably chosen a (usually a = 1). Construct the polynomial of degree N [illegible]

$$P^*(X^*) = P(a + X^*)$$

$$= P(a) + \frac{P'(a)}{1!} X^* + \frac{P''(a)}{2!} X^{*2} + \ldots + \frac{P^{(N)}(a)}{N!} X^{*N}$$

whose coefficients are calculated by the following theorem, given in [3, PP. 54-56], which repeatedly uses Horner's method to obtain the coefficients of the Taylor series expansion of P(X) about the point X = a.

Theorem 6.4. Given the finite sequence of numbers $b_1^{(1)}, b_2^{(1)}, \ldots, b_{N+1}^{(1)}$, and the number a, generate the sequences $\{b_n^{(K)}\}$ for K = 2,3,...,N+2 recursively by:

$$b_1^{(K)} = b_1^{(1)}$$

$$b_j^{(K)} = b_j^{(K-1)} + ab_{j-1}^{(K)}, \quad j = 2,3,\ldots,N+3-K.$$

Let $P_n(X) = b_1^{(1)}X^n + b_2^{(1)}X^{n-1} + \ldots + b_n^{(1)}X + b_{n+1}^{(1)}$ $(n = 0,1,\ldots,N)$. Then the numbers $b_n^{(K)}$ determined above satisfy

$$b_j^{(K)} = \frac{1}{(K-2)!} P_{j+K-3}^{(K-2)}(a), \quad j = 2,\ldots,N+3-K$$
$$K = 2,\ldots,N+2$$

where $P^{(0)}(a) = P(a)$.

The above theorem indicates the following array for N = 4.

$$\begin{array}{ccccc}
b_1^{(1)} & b_2^{(1)} & b_3^{(1)} & b_4^{(1)} & b_5^{(1)} \\
\hline
b_1^{(2)} & b_2^{(2)} & b_3^{(2)} & b_4^{(2)} & \underline{b_5^{(2)}} \\
b_1^{(3)} & b_2^{(3)} & b_3^{(3)} & \underline{b_4^{(3)}} & \\
b_1^{(4)} & b_2^{(4)} & \underline{b_3^{(4)}} & & \\
b_1^{(5)} & \underline{b_2^{(5)}} & & & \\
b_1^{(6)} & & & &
\end{array}$$

Figure 3.4. Scheme for Removing Zero Coefficients

The coefficients of the new polynomial $P^*(X^*)$ are the diagonal elements underlined in the scheme above; that is,

$$P^*(X^*) = b_1^{(6)}X^{*^4} + b_2^{(5)}X^{*^3} + b_3^{(4)}X^{*^2} + b_4^{(3)}X^* + b_5^{(2)}$$

where

$$P(X) = b_1^{(1)}X^4 + b_2^{(1)}X^3 + b_3^{(1)}X^2 + b_4^{(1)}X + b_5^{(1)}.$$

The coefficients of $P^*(X^*)$ will all be non-zero for sufficiently small values of a, since if a is not a zero of P(X) then the first N derivatives of P(X) exist and are not zero at X = a. Once the zeros, $\{X_K^*\}_{K=1}^4$, of $P^*(X^*)$ have been computed, the zeros, $\{X_K\}_{K=1}^4$, of P(X) are found from the formula $X_K = X_K^* + a$. As an example of the above theorem, let

$$P(X) = 12X^4 - 3X^3 + 7X + 10.$$

The coefficient of the X^2 term is 0, so the Q-D scheme cannot be started. With a = 1 we have for the above scheme

12	-3	0	7	10
12	9	9	16	26
12	21	30	46	
12	33	63		
12	45			
12				

Then

$$P^*(X^*) = 12X^{*4} + 45X^{*3} + 63X^{*2} + 46X + 26.$$

Let the zeros of $P^*(X^*)$ be $X_1^*, X_2^*, X_3^*, X_4^*$. Then the zeros of P(X) are $X_i = X_i^* + 1$, i = 1,2,3,4. This concludes the example.

All necessary cases have thus been considered. They are:

1. distinct zeros
2. zeros of equal magnitude
3. zero coefficients of the polynomial P(X).

Hence, the Quotient-Difference method can now be applied to the polynomial $P(X) = a_1X^N + a_2X^{N-1} + \dots + a_NX + a_{N+1}$ where $a_1 \neq 0 \neq a_{N+1}$.

6. Procedure for the Q-D Method

The sequence of basic steps performed by this method are as follows:

1. Given coefficients $a_1, a_2, \ldots, a_{N+1}$ of the N^{th} degree polynomial, $P(X)$, such that $a_1 \neq 0 \neq a_{N+1}$.
2. Determine if any of the coefficients in step 1 are equal to zero. If not, proceed to step 3. If one or more are zero, use Theorem 6.4 to construct a polynomial whose coefficients are all non-zero.
3. Begin the Q-D scheme by constructing the first two rows with the formulas:

$$q_1^{(1)} = -a_2/a_1$$

$$q_1^{(K)} = 0 \quad (K = 2,3,\ldots,N)$$

$$e_1^{(K)} = a_{K+1}\Big/a_K \quad (K = 2,3,\ldots,N).$$

Set the first and last columns to zero with

$$e_n^{(1)} = e_n^{(N+1)} = 0.$$

4. Construct the Q-D scheme row after row by using the rhombus rules:

$$q_{n+1}^{(K)} = \left(e_n^{(K+1)} - e_n^{(K)}\right) + q_n^{(K)} \quad (K = 1,2,\ldots,N)$$

$$e_{n+1}^{(K)} = \frac{q_{n+1}^{(K)}}{q_{n+1}^{(K-1)}}\, e_n^{(K)} \quad (K = 2,3,\ldots,N).$$

5. Test for convergence of the e-columns. Either of the following tests may be used:

a. $2|e_n^{(K)}| \Big/ (|q_n^{(K)} + q_n^{(K+1)}| + |q_{n-1}^{(K)} + q_{n-1}^{(K+1)}|)$

b. -EPS $|e_n^{(K)}| \,/\, |e_{25}^{(K)}|$ -EPS where EPS is chosen sufficiently small so that the test expression will remain positive until the K^{th} e-column has sufficiently converged. n is the number of iterations and is an input value.

6. After the maximum number of iterations, determine which e-columns have not converged. Then divide the Q-D scheme into subtables. Proceeding from left to right in the scheme, do step 7 or step 8 according to the number of q-columns in the subtable until all subtables have been considered.

7. If a subtable has two or more q-columsn in it, use Theorem 6.3 to determine the zeros of that subtable.

8. If a subtable contains only one q-column, then the final element in the q-column is the desired approximation of a zero of the polynomial.

9. Check the root count. If not all roots have been found, divide out the zeros that have been found by successively deflating P(X) using synthetic division. Shift the variable in the resulting polynomial for which no zeros were found by using Theorem 6.4. Then return to step 1. See Theorem 2.3 for the iteration formulas used to deflate a polynomial.

10. If it was necessary to use Theorem 6.4 in either step 2 or 9, the zeros $\{X_K^*\}_{K=1}^N$ found are the zeros of the new polynomial $P^*(X^*)$. Use $X_K = X_K^* + a$ to find the zeros $\{X_K\}_{K=1}^N$ of P(X).

7. Conclusion of the Method

As suggested earlier, it was decided to use the Q-D method for the purpose of producing good initial approximations to the zeros which would then be used with another method to produce final results. The following difficulty was encountered. In order to program the Q-D method to solve polynomials with maximum degree of twenty five, the size of core storage needed was considered and found to be quite large. For M roots of equal magnitude, Theorem 6.3 generates M sequences of polynomials having degrees 1,2,3,...,M. The number of terms needed in each sequence to produce convergence is not known before hand. In addition, M terms of the first sequence are necessary to compute the first term of the Mth sequence since, as seen from the scheme, the sequences are generated in a triangular manner. Therefore, in order to get sufficient convergence of the Mth sequence of polynomials, the number of terms in the first few sequences could become quite large. For example, suppose in a polynomial of degree 25, 20 roots have equal moduli. Then Theorem 6.3 calls for 20 sequences of polynomials to be generated, the first of degree 1, the second of degree 2, ..., and the 20th of degree 20. Twenty terms of the first, 19 terms of the second, and so on, would be necessary to calculate the first term of the 20th sequence. Thus,

$$\sum_{I=2}^{21} I(22 - I)$$

locations would be needed to store all coefficients. In general, for M roots of equal moduli,

$$\sum_{I=2}^{M+1} I(M + 2 - I)$$

locations are needed to compute the first term of the M^{th} sequence. An attempt was made to decrease the number of locations needed for storing the polynomial sequences by computing the terms of each polynomial sequence, storing the last polynomial of each sequence and repeating in order to continue each sequence using the same storage area for each repetition. However, this process of transferring polynomials, storing polynomials, bringing in new q-columns from the Q-D scheme, and continuing the polynomial sequences until convergence was obtained, presented a programming problem of greater magnitude than had been anticipated. Thus, considering the immediate purpose of this method, to give good initial approximations to the zeros, the slow convergence, and the lack of time, it was felt unprofitable to complete the programming of the Q-D method. Also, storage location requirements for the elements of the e-columns and q-columns of the Q-D scheme could become quite large unless the scheme is generated a few rows at a time. To do this would further add to the problem of generating the polynomial sequences using the q-columns.

CHAPTER VII

CONCLUSION

In order to compare Newton's, Muller's, and the greatest common divisor (g.c.d.) methods, consider the polynomials as being divided into the following classes:

1. polynomials with all distinct roots
2. polynomials with multiple roots
3. polynomials with roots close together

By "close together" I mean distinct roots agreeing in at least the first three significant digits. These roots will not be included in class 1.

The comparisons in the following material, except where specifically noted, are results of tests made on the IBM S/360 mod. 50 computer which has a 32 bit word. The programs were successfully run on the CDC 6600 and the UNIVAC 1108 which have a 60 bit word and a 36 bit word respectively. It was noted that the UNIVAC 1108 is about 15 times faster than the IBM S/360 mod. 50. The CDC 6600 is faster than the UNIVAC 1108 but the difference is not as great as that between the UNIVAC 1108 and the IBM S/360 mod. 50.

1. Polynomials With all Distinct Roots

First consider the class of polynomials having distinct roots. Newton's method is particularly suited for this class of polynomials. Its quadratic convergence is very fast which can save time and money to the user. The accuracy obtained is excellent as shown in exhibits A (12 sec.) and B (22 sec.) which present the zeros of a 15th degree polynomial in single precision and double precision, respectively. In most cases, the method produces convergence for almost any initial approximation given.

Muller's method also produces good results on this class of polynomials. The rate of convergence is, however, somewhat slower than Newton's method. This fact is especially significant when working with polynomials of high degree. The accuracy obtained by Muller's method is comparable to, but does not exceed that of Newton's method. In most cases, the accuracy of the two methods does not differ by more than one or two decimal places. Exhibits C (17 sec.) and D (28 sec.) show results of Muller's method for the polynomial of exhibits A and B. As in Newton's method, convergence is produced for almost any initial approximation given.

The g.c.d. method, whether used with Newton's or Muller's method as a supporting method on this class of polynomials, is no better than Newton's or Muller's method alone. The reason for this is that the greatest common divisor of the polynomial, P(X), and its derivative is 1. Then H(X) = P(X)/g.c.d. P(X) = P(X); that is, the polynomial solved by the supporting method is the same as the original polynomial. Thus, the g.c.d. method will require a considerable amount of extra time to perform useless calculations and will not produce better results than the supporting method used alone.

Thus, this class of polynomials presents no difficulty to any of these three methods. Newton's method, because of its speed, is therefore recommended.

2. Polynomials With Multiple Roots

Next consider the class of polynomials containing multiple roots. This class presents considerable difficulty for Newton's method, especially those polynomials containing roots of high multiplicity or containing a considerable number of multiple roots. The iteration formula for Newton's method is

$$X_{n+1} = X_n - P(X_n)/P'(X_n).$$

If c is a multiple root then $P(c) = P'(c) = 0$. Hence, as $X_n \rightarrow c$, $P(X_n) \rightarrow 0$ and $P'(X_n) \rightarrow 0$ and the iteration formula may be unstable, resulting in no convergence or bad accuracy. As the number of multiple roots increases, the polynomial becomes more ill-conditioned, convergence becomes more difficult, and accuracy is lost. Thus, the possibility of convergence decreases. This also holds true if the multiplicities of the roots are increased. The rate of convergence of Newton's method is much slower for multiple roots than for distinct roots. Exhibits E (38 sec.) and F (34 sec.) show a polynomial containing two multiple roots solved in single precision and double precision, respectively. Note the following from exhibit F:

1. Roots #2 and #3 are greatly improved by iterating on the original polynomial. Distinct roots are usually improved in this manner.
2. The time taken to solve this 6th degree equation with multiple roots is greater than the time taken by the same program to solve a 15th degree polynomial with all distinct roots (exhibit B).

3. Root #2 did not pass the convergence test after 200 iterations even though it was improved. This is probably due to the fact that the polynomial from which root 2 was extracted had only one multiple root but the original polynomial from which it was extracted the second time had two multiple roots; that is, the original polynomial is more ill-conditioned.
4. The accuracy of the roots before the attempt to improve accruacy is very poor. Root #2 is accurate to only three decimal places as compared to the 15 decimal places in exhibit B for distinct roots. Root #3 is especially bad, the imaginary part begin accurate to only one decimal place.

As seen from these exhibits, single precision calculations do not produce good results when multiple roots are involved. The time for exhibit E was 38 seconds. Compare this with exhibit A for the 15th degree polynomial. Exhibit G (14 sec.) is a polynomial containing two multiple roots. Note the poor results obtained before the attempt to improve accuracy and the improvement afterward.

In most cases involving roots of high multiplicity or several multiple roots, Newton's method fails to determine the correct multiplicity of many of the roots even though all the roots have been obtained.

In many cases, Newton's method fails to converge altogether. The polynomial with roots (the number in parentheses indicates multiplicity) 2 + 2i (3), 1 + .2i (2), -1 + .5i (3) could not be solved using Newton's method with a maximum number of 200 iterations and a convergence requirement of 10^{-10}. This polynomial is given as number 25 in exhibit H using Muller's method. Observe that this polynomial is polynomial #53 of exhibit F with the multiplicity of root -1 + .5i increased from 1 to 3.

Muller's method also encounters difficulty, although to a lesser degree than Newton's method, on this class of polynomials. In most cases, Muller's method produces convergence even when Newton's method completely fails. Newton's method completely failed for polynomial #25 but convergence was obtained using Muller's method as shown in exhibit H (40 sec.). The accuracy obtained by Muller's method is not good but usually better than Newton's method using the same convergence requirement. Compare exhibit J (15 sec.) with exhibit G. The rate of convergence of Muller's method is considerably slower for multiple roots than for distinct roots. However, for multiple roots, Muller's method is as fast or faster than Newton's. Newton's method appears to determine multiplicities more correctly than Muller's method. Note the following from exhibit J:

1. Roots #3, #4, and #5 are greatly improved by iterating on the original polynomial, especially the distinct root #4.
2. The accuracy obtained before the attempt to improve accruacy is very poor compared to the accuracy obtained with all distinct roots. The accuracy is, however, better than Newton's method (exhibit G).
3. The multiplicities of the roots are not determined correctly.

Polynomial #46 in single precision is given in exhibit I. Similarly as with Newton's method, an increase in the number of multiple roots or their multiplicities causes the polynomial to become more ill-conditioned. As a result, accuracy is lost.

The g.c.d. method is perfectly suited for polynomials with multiple roots. All multiple roots are removed leaving only a polynomial of class 1 (all distinct roots) to be solved. This indicates that best results should be obtained by using Newton's method as the supporting method, since Newton's method enjoys the advantage of speed over Muller's method for distinct roots. This has indeed proved to be true. The accuracy of the roots obtained decreases, somewhat, when the number of multiple roots is increased. This is due to accuracy lost in computing the g.c.d. and the quotient polynomial and not as a result of the supporting method. The accuracy obtained using each supporting method is about the same.

Multiplicities are determined with excellent accuracy. The g.c.d. method is not as sensitive to roots of high multiplicity or polynomials containing a large number of multiple roots as are both Newton's and Muller's. The g.c.d. method is faster than either Newton's or Muller's because multiple roots greatly slow the rate of convergence of the latter two. Exhibits K (2 sec.) and L (3 sec.) show polynomial #25 for which Newton's method gave no convergence and Muller's method gave poor convergence. Note that the execution time for double precision is considerably less than for Muller's method alone (exhibit H). The multiplicities are correct. Exhibit M (3 sec.) is a polynomial containing two roots each of multiplicity 6. Newton's method used alone did not produce convergence to any root of this polynomial and Muller's method gave bad accuracy. Note the accuracy and speed of the g.c.d. method on this polynomial. Exhibit N (12 sec.) contains 6 multiple roots. Observe that the accuracy is not quite as good as in the preceding two exhibits becasue there are more multiple roots. Note also that there is not much improvement in the roots after the attempt to improve accruacy. This is characteristic of Newton's and Muller's accuracy on distinct roots, and further supports the conclusion that the loss of accuracy is a result of computing the g.c.d. polynomial. Again the multiplicities are all correct.

The accuracy of multiple roots obtained by the CDC 6600 is considerably better than the IBM S/360 mod. 50 while that obtained by the UNIVAC 1108 is only a little better than the IBM S/360 mod. 50.

Therefore, for polynomials with multiple roots, the order in which the three methods are recommended, beginning with the best is: g.c.d. with Newton's, Muller's, Newton's.

Since multiple roots are obtained with less accuracy than distinct roots, lowering the convergence requirement produced convergence in many cases where the higher convergence requirement fails to produce convergence. The accuracy is usually fair for a polynomial with few multiple roots and decreases as the number of multiple roots increases. In most cases, the accuracy is sufficient to give a good approximation of the roots. Exhibit O shows polynomial #25 as a result of using the convergence requirement of 10^{-5}. The roots after the attempt to improve accuracy are accurate to at least 4 decimal places with some gaining accuracy to 6 or 7 decimal places. Recall that the convergence requirement of 10^{-10} produced no convergence.

3. Polynomials With Roots Close Together

Polynomials with distinct roots close together present considerable difficulty for all three methods. Ability to converge and the accuracy obtained depends on the number of these roots involved and their closeness. If care is not taken in choosing the multiplicity requirement, roots very close together may not be separated and, as a result, will be obtained as multiple roots. Exhibit P (5 sec.) shows what happens when the multiplicity requirement is too small. Note the following from this exhibit:

1. Root .103 + .103i was not obtained.
2. Root #2 has multiplicity 2 which is incorrect.
3. Root #3, before the attempt to improve accuracy, is entirely incorrect. This is the result of deflating by an incorrect root. The correct root was obtained by iterating on the original polynomial. This emphasizes the necessity of observing the results both before and after the attempt to improve accuracy.

When the results indicate that roots close together are present, the multiplicity requirement should be sufficiently increased to separate the roots. This was done on the above polynomial and the results and are indicated in exhibit Q (5 sec.). Note the reasonable accuracy of these roots all of which agree in the first two significant digits but differ in the third. Newton's method failed to converge to any root on polynomial #33 with roots, 1.010 + 1.020i, 1.011 + 1.021i, 1.012 + 1.022i, 1.013 + 1.023i within 200 iterations and using a convergence requirement of 10^{-10}. Note that these roots agree in the first three significant digits but differ in the fourth.

Polynomial #33 was solved with excellent accuracy by the CDC 6600. The test for convergence was 10^{-14} and the test for multiplicities was 10^{-8}. The roots, as found, were

```
Root (1) = 1.0100000000000000D+00 + 1.0200000000000000D+00 I
Root (2) = 1.0110000000000000D+00 + 1.0210000000000000D+00 I
Root (3) = 1.0130000000000000D+00 + 1.0230000000000000D+00 I
Root (4) = 1.0120000000000000D+00   1.0220000000000000D+00 I
```

This polynomial was also solved by the UNIVAC 1108 using a test for convergence of 10^{-8} and a test for multiplicities of 10^{-6}. The results were

```
Root (1) = .1010000000884303+001 + .1019999999482552+001 I
Root (2) = .1011000000380731+001 + .1020999998517433+001 I
Root (3) = .1012999998246209+001 + .1022999998494446+001 I
Root (4) = .1012000001356113+001 + .1021999999747005+001 I
```

Compare these results with exhibit V.

Muller's method is recommended for this class of polynomials because it produces convergence in most cases where Newton's method fails. When both methods produce convergence, the accuracy obtained by Muller's method is as good or better than that obtained by Newton's. Compare exhibits R (20 sec.) and Q. Again, a multiplicity requirement chosen too small produces incorrect results as in Newton's method. Exhibit S gives polynomial #33 solved by Muller's method. Observe that roots #2 and #4 obtained before the attempt to improve accuracy converged to the same root during the attempt to improve accuracy. Recall that Newton's method failed to converge to any root of this polynomial.

The g.c.d. method, as in the case of all distinct roots, is no better than the supporting method used. In many cases where very close roots are involved, the incorrect greatest common divisor may be obtained by treating these as multiple roots. Thus, incorrect results will be obtained. This occurs when the requirement check for zero given to the GCD routine is too low. This requirement can be increased to correct the error. However, if this requirement is made too high when multiple roots are involved, an incorrect g.c.d. may be obtained. Also, a multiplicity requirement too low will produce incorrect results. Exhibit T shows the result of both the epsilon check for zero in subroutine GCD and the epsilon check for multiplicities being too low. Both the root and the multiplicity are incorrect. These requirements were increased and the result given as exhibit U (7 sec.). Correct results were obtained. Thus, as a result of having to carefully choose both the epsilon check in subroutine GCD and the multiplicity requirement, this method is not recommended for this class of polynomials.

As in the case of multiple roots, the convergence requirement may be lowered to obtain convergence in most cases where a high convergence requirement causes failure to converge. Convergence was obtained for polynomial #33 (exhibit V (4 sec.)) by lowering the convergence requirement from 10^{-10} to 10^{-8}.

Another technique for obtaining convergence on ill-conditioned polynomials is to add some distinct roots which are not close together. This is accomplished by multiplying the polynomial by several linear factors. As an example of this technique, consider polynomial #33 on which Newton's method failed to converge. After multiplying the polynomial by the factors (X - 4 + 2i) and (X - 3 - 5i), convergence was obtained as indicated in exhibit W (24 sec.). The number of distinct roots added to obtain convergence depends on how bad the polynomial is ill-conditioned.

For polynomials containing a combination of these three types of roots, the method used to obtain best accuracy depends on the number of each type involved. The g.c.d. method is usually best because distinct and multiple roots are obtained with good accuracy. Those roots obtained with poor accuracy are probably roots very close together. Increasing the test for zero in subroutine GCD may separate these close roots but multiple roots may not be obtained with good accuracy. If roots close together are present, another method may be used to separate these roots. Exhibit X (6 sec.) gives a polynomial with two multiple roots, two distinct roots not close together, and two distinct roots close together. This polynomial was first solved using the g.c.d. method with Newton's method as the supporting method. Note that all roots except the roots close together are reasonably accurate and their multiplicities correct. Using the results after the attempt to improve accuracy as initial approximations, this polynomial was solved by Muller's method as indicated in exhibit Y (39 sec.). The multiplicity requirement was increased considerably above normal to prevent any root, especially roots very close together, from being obtained as multiple roots. This tends to separate the roots close together. Note the following:

1. The two roots close together were obtained very accurately before the attempt to improve accuracy but both converged to the same root later.
2. The accuracy of all roots is good.

Exhibit Z (18 sec.) presents the results of solving this polynomial by Muller's method but letting the program generate its own initial approximations. Note that the accuracy is not as good as that obtained using the initial approximations close to the roots.

Thus, each method is particularly suited for a different type of polynomial. Newton's is best for polynomials with distinct roots not too close together. The g.c.d. method is superior for polynomials having multiple roots, while Muller's method is recommended for extracting roots very close together.

```
          NEWTONS METHOD TO FIND ZEROS OF POLYNOMIALS
          POLYNOMIAL NUMBER  7 OF DEGREE 15


THE COEFFICIENTS OF P(X) ARE

   P( 1) =  0.3000000E 01 +  0.0000000E 00 I
   P( 2) = -0.1789999E 02 +  0.0000000E 00 I
   P( 3) =  0.2010001E 02 + -0.6575999E 02 I
   P( 4) =  0.1745000E 03 +  0.2843679E 03 I
   P( 5) = -0.7594199E 03 +  0.3118081E 03 I
   P( 6) =  0.8274360E 03 + -0.3069040E 04 I
   P( 7) =  0.1329084E 04 +  0.4710625E 04 I
   P( 8) = -0.5611859E 04 + -0.1674576E 04 I
   P( 9) =  0.7224758E 04 + -0.1548288E 04 I
   P(10) = -0.2276992E 04 +  0.3046320E 04 I
   P(11) = -0.1241472E 04 + -0.4097281E 04 I
   P(12) =  0.5402801E 04 +  0.1263488E 04 I
   P(13) = -0.6468336E 04 +  0.1236480E 04 I
   P(14) =  0.1467456E 04 +  0.2272000E 02 I
   P(15) = -0.1077120E 03 + -0.5475840E 03 I
   P(16) =  0.3456000E 02 +  0.1267200E 03 I


NUMBER OF INITIAL APPROXIMATIONS GIVEN.   0
MAXIMUM NUMBER OF ITERATIONS.           200
TEST FOR CONVERGENCE.              0.10E-04
TEST FOR MULTIPLICITIES.           0.10E-01
RADIUS TO START SEARCH.            0.00E 00
RADIUS TO END SEARCH.              0.00E 00


BEFORE THE ATTEMPT TO IMPROVE ACCURACY, THE ZEROS OF P(X) ARE


ZEROS OF P(X)                                        MULTIPLICITIES           INITIAL APPROXIMATION

   ROOT( 1) =  0.3000001E 00 +  0.6291682E-08 I            1            0.4829629E 00 +  0.1294095E 00 I
   ROOT( 2) =  0.2000000E 00 + -0.2000002E 00 I            1            0.7071067E 00 +  0.7071068E 00 I
   ROOT( 3) = -0.2000002E 00 +  0.2000002E 00 I            1            0.3882295E 00 +  0.1448888E 01 I
   ROOT( 4) =  0.1789849E-06 +  0.1000000E 01 I            1           -0.5176364E 00 +  0.1931851E 01 I
   ROOT( 5) = -0.1000000E 01 + -0.3287970E-06 I            1           -0.1767765E 01 +  0.1767768E 01 I
   ROOT( 6) = -0.1000000E 01 + -0.1000002E 01 I            1           -0.2897776E 01 +  0.7764602E 00 I
   ROOT( 7) = -0.2310580E-06 + -0.1000000E 01 I            1           -0.3380741E 01 + -0.9058627E 00 I
   ROOT( 8) = -0.1999997E 01 + -0.3000000E 01 I            1           -0.2828430E 01 + -0.2828424E 01 I
   ROOT( 9) =  0.9999008E 00 + -0.2495819E-03 I            1           -0.1164691E 01 + -0.4346664E 01 I
   ROOT(10) =  0.2000046E 01 + -0.1000127E 01 I            1            0.1294087E 01 + -0.4829631E 01 I
   ROOT(11) =  0.2000143E 01 +  0.3578651E-03 I            1            0.3889081E 01 + -0.3889092E 01 I
   ROOT(12) =  0.2999989E 01 + -0.5277649E-04 I            1            0.5795551E 01 + -0.1552923E 01 I
   ROOT(13) =  0.3999991E 01 +  0.4000009E 01 I            1            0.6278520E 01 +  0.1682312E 01 I
   ROOT(14) =  0.9999154E 00 +  0.1000066E 01 I            1            SOLVED BY DIRECT METHOD
   ROOT(15) = -0.3333335E 01 + -0.1112619E-05 I            1            SOLVED BY DIRECT METHOD


AFTER  THE ATTEMPT TO IMPROVE ACCURACY, THE ZEROS OF P(X) ARE
```

Exhibit A.

```
ZEROS OF P(X)                                          MULTIPLICITIES          INITIAL APPROXIMATION

  ROOT( 1) =  0.3000001E 00 + -0.9981129E-08 I               1             0.4829629E 00 +  0.1294095E 00 I
  ROOT( 2) =  0.2000000E 00 + -0.2000000E 00 I               1             0.7071067E 00 +  0.7071068E 00 1
  ROOT( 3) = -0.2000000E 00 +  0.2000000E 00 I               1             0.3882295E 00 +  0.1448888E 01 I
  ROOT( 4) =  0.1446784E-07 +  0.1000000E 01 I               1            -0.5176364E 00 +  0.1931851E 01 I
  ROOT( 5) = -0.1000000E 01 + -0.2083993E-08 I               1            -0.1767765E 01 +  0.1767768E 01 I
  ROOT( 6) = -0.1000000E 01 + -0.1000000E 01 I               1            -0.2897776E 01 +  0.7764602E 00 I
  ROOT( 7) =  0.1241104E-06 + -0.1000000E 01 I               1            -0.3380741E 01 + -0.9058627E 00 I
  ROOT( 8) = -0.2000000E 01 + -0.3000000E 01 I               1            -0.2828430E 01 + -0.2828424E 01 I
  ROOT( 9) =  0.1000000E 01 +  0.1584821E-06 I               1            -0.1164691E 01 + -0.4346664E 01 I
  ROOT(10) =  0.1999998E 01 + -0.1000001E 01 I               1             0.1294087E 01 + -0.4829631E 01 I
  ROOT(11) =  0.2000008E 01 +  0.8055767E-06 I               1             0.3889081E 01 + -0.3889092E 01 I
  ROOT(12) =  0.2999995E 01 +  0.2530969E-05 I               1             0.5795551E 01 + -0.1552923E 01 I
  ROOT(13) =  0.4000000E 01 +  0.4000002E 01 I               1             0.6278520E 01 +  0.1682312E 01 I
  ROOT(14) =  0.9999998E 00 +  0.1000000E 01 I               1             SOLVED BY DIRECT METHOD
  ROOT(15) = -0.3333334E 01 +  0.4381855E-06 I               1             SOLVED BY DIRECT METHOD
```

Exhibit A. Roots Are: -1 - i, 1 + i, -2 - 3i, 2 - i, 3, 2, i, -i, -10/3, .3, -1, 1, 4 + 4i, -.2 + .2i, .2 - .2i.

```
        NEWTONS METHOD TO FIND ZEROS OF POLYNOMIALS
        POLYNOMIAL NUMBER  7 OF DEGREE 15


THE COEFFICIENTS OF P(X) ARE

  P( 1) =  0.3000000000000000D 01 +  0.0000000000000000D 00 I
  P( 2) = -0.1790000000000000 02 + -0.0000000000000000D 00 I
  P( 3) =  0.2010000000000000 02 + -0.6576000000000001 02 I
  P( 4) =  0.1745000000000000 03 +  0.2843680000000000D 03 I
  P( 5) = -0.7594200000000000 03 +  0.3118080000000000 03 I
  P( 6) =  0.8274360000000000D 03 + -0.3069040000000000D 04 I
  P( 7) =  0.1329084000000000D 04 +  0.4710624000000000D 04 I
  P( 8) = -0.5611860000000000D 04 + -0.1674576000000000D 04 I
  P( 9) =  0.7224756000000000D 04 + -0.1548288000000000D 04 I
  P(10) = -0.2276992000000000D 04 +  0.3046320000000000D 04 I
  P(11) = -0.1241472000000000D 04 + -0.4097280000000000D 04 I
  P(12) =  0.5402800000000001D 04 +  0.1263488000000000D 04 I
  P(13) = -0.6468336000000001D 04 +  0.1236480000000000D 04 I
  P(14) =  0.1467456000000000D 04 +  0.2272000000000000D 02 I
  P(15) = -0.1077120000000000D 03 + -0.5475840000000000D 03 I
  P(16) =  0.3456000000000000D 02 +  0.1267200000000000D 03 I


NUMBER OF INITIAL APPROXIMATIONS GIVEN.   0
MAXIMUM NUMBER OF ITERATIONS.           200
TEST FOR CONVERGENCE.              0.10D-09
TEST FOR MULTIPLICITIES.           0.10D-01
RADIUS TO START SEARCH.            0.00D 00
RADIUS TO END SEARCH.              0.00D 00


BEFORE THE ATTEMPT TO IMPROVE ACCURACY, THE ZEROS OF P(X) ARE


 ROOTS OF P(X)                                                          MULTIPLICITIES                 INITIAL APPROXIMATION


  ROOT( 1) =  0.3000000000000000D 00 +  0.1609358115166531D-16 I            1          0.4829629115656279D 00 +  0.1294095284438187D 00 I
  ROOT( 2) =  0.2000000000000000D 00 + -0.2000000000000000D 00 I            1          0.7071067553046346D 00 +  0.7071068070684595D 00 I
  ROOT( 3) = -0.2000000000000000D 00 +  0.2000000000000000D 00 I            1          0.3882284792654056D 00 +  0.1448888763117193D 01 I
  ROOT( 4) =  0.2485026109655803D-17 +  0.1000000000000000 01 I             1         -0.5176382551966724D 00 +  0.1931851608368755D 01 I
  ROOT( 5) = -0.1000000000000000D 01 + -0.5321953706459946D-16 I            1         -0.1767767147080701D 01 +  0.1767766758852015D 01 I
  ROOT( 6) = -0.1000000000000000D 01 + -0.9999999999999998D 00 I            1         -0.2897777583074990D 01 +  0.7764567463987070D 00 I
  ROOT( 7) = -0.3106918759280349D-15 + -0.9999999999999994D 00 I            1         -0.3380740248331229D 01 + -0.9058671940816160D 00 I
  ROOT( 8) = -0.2000000000000000D 01 + -0.3000000000000000D 01 I            1         -0.2828426607107896D 01 + -0.2828427642384390D 01 I
  ROOT( 9) =  0.1000000000000016D 01 + -0.1009764996352100D-12 I            1         -0.1164684801399899D 01 + -0.4346666459873368D 01 I
  ROOT(10) =  0.2000000000000040D 01 + -0.1000000000000032D 01 I            1          0.1294096345098645D 01 + -0.4829628831453027D 01 I
  ROOT(11) =  0.1999999999999980D 01 +  0.1385507847222202D-12 I            1          0.3889088292979509D 01 + -0.3889086300072258D 01 I
  ROOT(12) =  0.3000000000000001D 01 + -0.1417549157449219D-13 I            1          0.5795555393512303D 01 + -0.1552912644268974D 01 I
  ROOT(13) =  0.3999999999999998D 01 +  0.4000000000000003D 01 I            1          0.6278517357734125D 01 +  0.1682325708247752D 01 I
  ROOT(14) =  0.9999999999996060D 00 +  0.1000000000000005D 01 I            1          SOLVED BY DIRECT METHOD
  ROOT(15) = -0.3333333333333334D 01 + -0.1850371707708594D-15 I            1          SOLVED BY DIRECT METHOD
```

Exhibit B.

```
AFTER  THE ATTEMPT TO IMPROVE ACCURACY, THE ZEROS OF P(X) ARE


 ROOTS OF P(X)                                                     MULTIPLICITIES              INITIAL APPROXIMATION


  ROOT( 1) =  0.30000000000000000D 00 +  0.33289435377059130D-16 I       1       0.48296291156562790D 00 +  0.12940952844381870D 00 I
  ROOT( 2) =  0.20000000000000000D 00 + -0.20000000000000000D 00 I       1       0.70710675530463460D 00 +  0.70710680706845950D 00 I
  ROOT( 3) = -0.20000000000000000 00 +  0.20000000000000000D 00 I       1       0.38822847926540560 00 +  0.14488887631171930D 01 I
  ROOT( 4) = -0.11260430437627530D-17 +  0.10000000000000000D 01 I       1      -0.51763825519667240D 00 +  0.19318516083687550D 01 I
  ROOT( 5) = -0.10000000000000000 01 + -0.18824780305507960D-16 I       1      -0.17677671470807010D 01 +  0.17677667588520150D 01 I
  ROOT( 6) = -0.10000000000000000D 01 + -0.10000000000000000D 01 I       1      -0.28977775830749900D 01 +  0.77645674639870700 00 I
  ROOT( 7) =  0.61318693193791400D-16 + -0.10000000000000000D 01 I       1      -0.33807402483312290D 01 + -0.90586719408161600D 00 I
  ROOT( 8) = -0.20000000000000000D 01 + -0.30000000000000000D 01 I       1      -0.28284266071078960D 01 + -0.28284276423843900D 01 I
  ROOT( 9) =  0.10000000000000000 01 +  0.72877654808950640D-16 I       1      -0.11646848013998990D 01 + -0.43466664598733680 01 I
  ROOT(10) =  0.20000000000000010 01 + -0.99999999999999990D 00 I       1       0.12940963450986450D 01 + -0.48296288314530270 01 I
  ROOT(11) =  0.20000000000000030D 01 + -0.39680396208113570D-14 I       1       0.38890882929795090D 01 + -0.38890863000722580 01 I
  ROOT(12) =  0.29999999999999970 01 +  0.28968896304378480-14 I       1       0.57955553935123030 01 + -0.15529126442689740D 01 I
  ROOT(13) =  0.40000000000000000 01 +  0.40000000000000010 01 I       1       0.62785173577341250 01 +  0.16823257082477520 01 I
  ROOT(14) =  0.99999999999999950 00 +  0.10000000000000000D 01 I       1       SOLVED BY DIRECT METHOD
  ROOT(15) = -0.33333333333333340 01 + -0.89980636705176380-16 I       1       SOLVED BY DIRECT METHOD
```

Exhibit B. Roots Are: -1 - i, 1 + i, -2 - 3i, 2 - i, 3, 2, i, -i, -10/3, .3, -1, 1, 4 + 4i, -.2 + .2i, .2 - .2i.

```
 MULLERS METHOD FOR FINDING THE ZEROS OF A POLYNOMIAL
 POLYNOMIAL NUMBER  7 OF DEGREE 15


 THE COEFFICIENTS OF P(X) ARE


  P( 1) =  0.3000000E 01 +  0.0000000E 00 I
  P( 2) = -0.1789999E 02 +  0.0000000E 00 I
  P( 3) =  0.2010001E 02 + -0.6575999E 02 I
  P( 4) =  0.1745000E 03 +  0.2843679E 03 I
  P( 5) = -0.7594199E 03 +  0.3118081E 03 I
  P( 6) =  0.8274360E 03 + -0.3069040E 04 I
  P( 7) =  0.1329084E 04 +  0.4710625E 04 I
  P( 8) = -0.5611859E 04 + -0.1674576E 04 I
  P( 9) =  0.7224758E 04 + -0.1548288E 04 I
  P(I0) = -0.2276992E 04 +  0.3046320E 04 I
  P(11) = -0.1241472E 04 + -0.4097281E 04 I
  P(12) =  0.5402801E 04 +  0.1263488E 04 I
  P(13) = -0.6468336E 04 +  0.1236480E 04 I
  P(14) =  0.1467456E 04 +  0.2272000E 02 I
  P(15) = -0.1077120E 03 + -0.5475840E 03 I
  P(16) =  0.3456000E 02 +  0.1267200E 03 I


NUMBER OF INITIAL APPROXIMATIONS GIVEN.   0
MAXIMUM NUMBER OF ITERATIONS.           200
TEST FOR CONVERGENCE.              0.10E-04
TEST FOR MULTIPLICITIES.           0.10E-01
RADIUS TO START SEARCH.            0.00E 00
RADIUS TO END SEARCH.              0.00E 00


BEFORE ATTEMPT TO IMPROVE ACCURACY


ROOTS OF P(X)                                         MULTIPLICITIES          INITIAL APPROXIMATION


 ROOT( 1) =  0.3000001E 00 +  0.2621618E-08 I              1           0.4829629E 00 +  0.1294095E 00 I
 ROOT( 2) = -0.2000000E 00 +  0.2000000E 00 I              1           0.7071067E 00 +  0.7071068E 00 I
 ROOT( 3) =  0.9999995E 00 +  0.1000000E 01 I              1           0.3882295E 00 +  0.1448888E 01 I
 ROOT( 4) =  0.9999982E 00 + -0.3130342E-05 I              1          -0.5176364E 00 +  0.1931851E 01 I
 ROOT( 5) = -0.1000000E 01 + -0.1000000E 01 I              1          -0.1767765E 01 +  0.1767768E 01 I
 ROOT( 6) = -0.3333335E 01 + -0.2337438E-06 I              1          -0.2897776E 01 +  0.7764602E 00 I
 ROOT( 7) = -0.1000060E 01 + -0.2979267E-04 I              1          -0.3380741E 01 + -0.9058627E 00 I
 ROOT( 8) = -0.2000000E 01 + -0.3000000E 01 I              1          -0.2828430E 01 + -0.2828424E 01 I
 ROOT( 9) =  0.2000399E 01 + -0.2037037E-03 I              1          -0.1164691E 01 + -0.4346664E 01 I
 ROOT(10) =  0.2999865E 01 + -0.7479227E-04 I              1           0.1294087E 01 + -0.4829631E 01 I
 ROOT(11) =  0.1622662E-04 +  0.9999509E 00 I              1           0.3889081E 01 + -0.3889092E 01 I
 ROOT(12) = -0.2123728E-03 + -0.1000231E 01 I              1           0.5795551E 01 + -0.1552923E 01 I
 ROOT(13) =  0.3999983E 01 +  0.4000027E 01 I              1           0.6278520E 01 +  0.1682312E 01 I
 ROOT(14) =  0.1999767E 01 + -0.1000011E 01 I              1           SOLVED BY DIRECT METHOD
 ROOT(15) =  0.2002156E 00 + -0.1994236E 00 I              1           SOLVED BY DIRECT METHOD
```

Exhibit C.

```
AFTER THE ATTEMPT TO IMPROVE ACCURACY

ROOTS OF P(X)                                            MULTIPLICITIES          INITIAL APPROXIMATION

 ROOT( 1) =  0.3000001E 00 +  0.2925553E-09 I                  1             0.4829629E 00 +  0.1294095E 00 I
 ROOT( 2) = -0.2000000E 00 +  0.2000000E 00 I                  1             0.7071067E 00 +  0.7071068E 00 I
 ROOT( 3) =  0.9999996E 00 +  0.1000000E 01 I                  1             0.3882295E 00 +  0.1448888E 01 I
 ROOT( 4) =  0.9999995E 00 + -0.4528033E-08 I                  1            -0.5176364E 00 +  0.1931851E 01 I
 ROOT( 5) = -0.1000002E 01 + -0.1000001E 01 I                  1            -0.1767765E 01 +  0.1767768E 01 I
 ROOT( 6) = -0.3333334E 01 +  0.1599741E-06 I                  1            -0.2897776E 01 +  0.7764602E 00 I
 ROOT( 7) = -0.1000000E 01 +  0.8811071E-08 I                  1            -0.3380741E 01 + -0.9058627E 00 I
 ROOT( 8) = -0.1999999E 01 + -0.3000000E 01 I                  1            -0.2828430E 01 + -0.2828424E 01 I
 ROOT( 9) =  0.2000004E 01 +  0.3921901E-06 I                  1            -0.1164691E 01 + -0.4346664E 01 I
 ROOT(10) =  0.2999994E 01 +  0.4597451E-06 I                  1             0.1294087E 01 + -0.4829631E 01 I
 ROOT(11) =  0.4144528E-07 +  0.1000000E 01 I                  1             0.3889081E 01 + -0.3889092E 01 I
 ROOT(12) =  0.1809455E-06 + -0.1000000E 01 I                  1             0.5795551E 01 + -0.1552923E 01 I
 ROOT(13) =  0.3999998E 01 +  0.4000003E 01 I                  1             0.6278520E 01 +  0.1682312E 01 I
 ROOT(14) =  0.2000000E 01 + -0.1000001E 01 I                  1             SOLVED BY DIRECT METHOD
 ROOT(15) =  0.2000000E 00 + -0.2000000E 00 I                  1             SOLVED BY DIRECT METHOD
```

Exhibit C. Roots Are: -1 - i, 1 + i, -2 - 3i, 2 - i, 3, 2, i, -i, -10/3, .3, -1, 1, 4 + 4i, -.2 + .2i, .2 - .2i.

```
MULLERS METHOD FOR FINDING THE ZEROS OF A POLYNOMIAL
POLYNOMIAL NUMBER  7 OF DEGREE 15

THE COEFFICIENTS OF P(X) ARE

 P( 1) =  0.30000000000000000D 01 +  0.00000000000000000D 00 I
 P( 2) = -0.17900000000000000D 02 + -0.00000000000000000D 00 I
 P( 3) =  0.20100000000000000D 02 + -0.65760000000000001D 02 I
 P( 4) =  0.17450000000000000D 03 +  0.28436800000000000D 03 I
 P( 5) = -0.75942000000000000D 03 +  0.31180800000000000D 03 I
 P( 6) =  0.82743600000000000D 03 + -0.30690400000000000D 04 I
 P( 7) =  0.13290840000000000D 04 +  0.47106240000000000D 04 I
 P( 8) = -0.56118600000000000D 04 + -0.16745760000000000D 04 I
 P( 9) =  0.72247560000000000D 04 + -0.15482880000000000D 04 I
 P(10) = -0.22769920000000000D 04 +  0.30463200000000000D 04 I
 P(11) = -0.12414720000000000D 04 + -0.40972800000000000D 04 I
 P(12) =  0.54028000000000001D 04 +  0.12634880000000000D 04 I
 P(13) = -0.64683360000000001D 04 +  0.12364800000000000D 04 I
 P(14) =  0.14674560000000000D 04 +  0.22720000000000000D 02 I
 P(15) = -0.10771200000000000D 03 + -0.54758400000000000D 03 I
 P(16) =  0.34560000000000000D 02 +  0.12672000000000000D 03 I


NUMBER OF INITIAL APPROXIMATIONS GIVEN.    0
MAXIMUM NUMBER OF ITERATIONS.            200
TEST FOR CONVERGENCE.                0.10D-09
TEST FOR MULTIPLICITIES.             0.10D-01
RADIUS TO START SEARCH.              0.00D 00
RADIUS TO END SEARCH.                0.00D 00


BEFORE ATTEMPT TO IMPROVE ACCURACY


ROOTS OF P(X)                                                     MULTIPLICITIES                    INITIAL APPROXIMATION

 ROOT( 1) =  0.30000000000000000D 00 +  0.33839987517518660D-16 I        1          0.4829629115656279D 00 +  0.1294095284438187D 00 I
 ROOT( 2) =  0.20000000000000000D 00 + -0.20000000000000000D 00 I        1          0.7071067553046346D 00 +  0.7071068070684595D 00 I
 ROOT( 3) =  0.99999999999999995D 00 +  0.99999999999999997D 00 I        1          0.3882284792654056D 00 +  0.1448888763117193D 01 I
 ROOT( 4) = -0.23890824083133820D-16 +  0.10000000000000000D 01 I        1         -0.5176382551966724D 00 +  0.1931851608368755D 01 I
 ROOT( 5) = -0.10000000000000000D 01 +  0.25316292776176400D-15 I        1         -0.1767767147080701D 01 +  0.1767766758852015D 01 I
 ROOT( 6) = -0.33333333333333334D 01 + -0.21447226690877370D-15 I        1         -0.2897777583074990D 01 +  0.7764567463987070D 00 I
 ROOT( 7) = -0.10000000000000008D 01 + -0.99999999999999661D 00 I        1         -0.3380740248331229D 01 + -0.9058671940816160D 00 I
 ROOT( 8) = -0.20000000000000001D 01 + -0.30000000000000000D 01 I        1         -0.2828426607107896D 01 + -0.2828427642384390D 01 I
 ROOT( 9) = -0.11388167309366500D-12 + -0.10000000000000109D 01 I        1         -0.1164684801399899D 01 + -0.4346666459873368D 01 I
 ROOT(10) =  0.19999999999998790D 01 + -0.99999999999999954D 00 I        1          0.1294096345098645D 01 + -0.4829628831453027D 01 I
 ROOT(11) =  0.29999999999999946D 01 +  0.19218912044391240D-14 I        1          0.3889088292979509D 01 + -0.3889086300072258D 01 I
 ROOT(12) =  0.20000000000000122D 01 + -0.22541548795801700D-12 I        1          0.5795555393512303D 01 + -0.1552912644268974D 01 I
 ROOT(13) =  0.39999999999999999D 01 +  0.40000000000000005D 01 I        1          0.6278517357734125D 01 +  0.1682325708247752D 01 I
 ROOT(14) =  0.10000000000000208D 01 +  0.22974446419373370D-12 I        1          SOLVED BY DIRECT METHOD
 ROOT(15) = -0.20000000000000366D 00 +  0.20000000000000595D 00 I        1          SOLVED BY DIRECT METHOD
```

Exhibit D.

```
AFTER THE ATTEMPT TO IMPROVE ACCURACY


ROOTS OF P(X)                                                MULTIPLICITIES              INITIAL APPROXIMATION


 ROOT( 1) =  0.30000000000000000 00 +  0.33767394620832440-16 I        1          0.48296291156562790 00 +  0.12940952844381870 00 I
 ROOT( 2) =  0.20000000000000000 00 + -0.20000000000000000 00 I        1          0.70710675530463460 00 +  0.70710680706845950 00 I
 ROOT( 3) =  0.99999999999999950 00 +  0.10000000000000000 01 I        1          0.38822847926540560 00 +  0.14488887631171930 01 I
 ROOT( 4) =  0.11537757751536680-16 +  0.10000000000000000 01 I        1         -0.51763825519667240 00 +  0.19318516083687550 01 I
 ROOT( 5) = -0.10000000000000000 01 + -0.99356260300168480-17 I        1         -0.17677671470807010 01 +  0.17677667588520150 01 I
 ROOT( 6) = -0.33333333333333340 01 + -0.19191253971536640-15 I        1         -0.28977775830749900 01 +  0.77645674639870700 00 I
 ROOT( 7) = -0.10000000000000000 01 + -0.99999999999999980 00 I        1         -0.33807402483317290 01 + -0.90586719408161600 00 I
 ROOT( 8) = -0.20000000000000000 01 + -0.30000000000000000 01 I        1         -0.28284266071078960 01 + -0.28284276423843900 01 I
 ROOT( 9) =  0.55291352338277240-16 + -0.10000000000000000 01 I        1         -0.11646848013998990 01 + -0.43466664598733680 01 I
 ROOT(10) =  0.20000000000000020 01 + -0.99999999999999860 00 I        1          0.12940963450986450 01 + -0.48296288314530270 01 I
 ROOT(11) =  0.29999999999999960 01 +  0.24394773729111440-14 I        1          0.38890882929795090 01 + -0.38890863000722580 01 I
 ROOT(12) =  0.20000000000000020 01 + -0.41020504900166570-14 I        1          0.57955553935123030 01 + -0.15529126442689740 01 I
 ROOT(13) =  0.40000000000000000 01 +  0.40000000000000010 01 I        1          0.62785173577341250 01 +  0.16823257082477520 01 I
 ROOT(14) =  0.10000000000000000 01 +  0.76824382838570700-16 I        1          SOLVED BY DIRECT METHOD
 ROOT(15) = -0.20000000000000000 00 +  0.20000000000000000 00 I        1          SOLVED BY DIRECT METHOD
```

Exhibit D. Roots Are: -1 - i, 1 + i, -2 - 3i, 2 - i, 3, 2, i, -i, -10/3, .3, -1, 1, 4 + 4i, -.2 +.2i, .2 - .2i.

```
     NEWTONS METHOD TO FIND ZEROS OF POLYNOMIALS
     POLYNOMIAL NUMBER 53 OF DEGREE  6


THE COEFFICIENTS OF P(X) ARE

  P( 1) =  0.1000000E 01 +  0.0000000E 00 I
  P( 2) = -0.7000000E 01 + -0.1050000E 02 I
  P( 3) = -0.2800000E 02 +  0.5800000E 02 I
  P( 4) =  0.1710000E 03 +  0.1500000E 01 I
  P( 5) = -0.7300000E 02 + -0.2510000E 03 I
  P( 6) = -0.2280000E 03 +  0.1040000E 03 I
  P( 7) =  0.7200000E 02 +  0.1040000E 03 I


NUMBER OF INITIAL APPROXIMATIONS GIVEN.   0
MAXIMUM NUMBER OF ITERATIONS.           200
TEST FOR CONVERGENCE.              0.10E-04
TEST FOR MULTIPLICITIES.           0.10E-01
RADIUS TO START SEARCH.            0.00E 00
RADIUS TO END SEARCH.              0.00E 00


BEFORE THE ATTEMPT TO IMPROVE ACCURACY, THE ZEROS OF P(X) ARE


ZEROS OF P(X)                                       MULTIPLICITIES          INITIAL APPROXIMATION

  ROOT( 1) =  0.9940745E 00 +  0.2009475E 01 I            2             0.4829629E 00 +  0.1294095E 00 I
  ROOT( 2) =  0.1814509E 01 +  0.1788034E 01 I            1             0.7071067E 00 +  0.7071068E 00 I
  ROOT( 3) =  0.2229694E 01 +  0.1938903E 01 I            1             0.3882295E 00 +  0.1448888E 01 I
  ROOT( 4) =  0.1968111E 01 +  0.2254501E 01 I            1             SOLVED BY DIRECT METHOD
  ROOT( 5) = -0.1000464E 01 +  0.4996119E 00 I            1             SOLVED BY DIRECT METHOD

IN THE ATTEMPT TO IMPROVE ACCURACY, ROOT( 2) =  0.1814509E 01 +  0.1788034E 01 I DID NOT CONVERGE.
THE PRESENT APPROXIMATION AFTER 200 ITERATIONS IS PRINTED BELOW.

IN THE ATTEMPT TO IMPROVE ACCURACY, ROOT( 3) =  0.2229694E 01 +  0.1938903E 01 I DID NOT CONVERGE.
THE PRESENT APPROXIMATION AFTER 200 ITERATIONS IS PRINTED BELOW.

IN THE ATTEMPT TO IMPROVE ACCURACY, ROOT( 4) =  0.1968111E 01 +  0.2254501E 01 I DID NOT CONVERGE.
THE PRESENT APPROXIMATION AFTER 200 ITERATIONS IS PRINTED BELOW.


AFTER  THE ATTEMPT TO IMPROVE ACCURACY, THE ZEROS OF P(X) ARE


ZEROS OF P(X)                                       MULTIPLICITIES          INITIAL APPROXIMATION

  ROOT( 1) =  0.9940745E 00 +  0.2009475E 01 I            2             0.4829629E 00 +  0.1294095E 00 I
  ROOT( 2) =  0.2049402E 01 +  0.2042129E 01 I            1             0.7071067E 00 +  0.7071068E 00 I
  ROOT( 3) =  0.1937237E 01 +  0.2044566E 01 I            1             0.3882295E 00 +  0.1448888E 01 I
  ROOT( 4) =  0.1941931E 01 +  0.2030631E 01 I            1             SOLVED BY DIRECT METHOD
  ROOT( 5) = -0.9999999E 00 +  0.4999999E 00 I            1             SOLVED BY DIRECT METHOD
```

Exhibit E. Roots Are: 2 + 2i (3), 1 + 2i (2), -1 + .5i (1).

```
      NEWTONS METHOD TO FIND ZEROS OF POLYNOMIALS
      POLYNOMIAL NUMBER 53 OF DEGREE  6


THE COEFFICIENTS OF P(X) ARE

  P( 1) =  0.1000000000000000D 01 +  0.0000000000000000 00 I
  P( 2) = -0.7000000000000001D 01 + -0.1050000000000000D 02 I
  P( 3) = -0.2800000000000000D 02 +  0.5800000000000001D 02 I
  P( 4) =  0.1710000000000000D 03 +  0.1500000000000000D 01 I
  P( 5) = -0.7300000000000000D 02 + -0.2510000000000000D 03 I
  P( 6) = -0.2280000000000000D 03 +  0.1040000000000000 03 I
  P( 7) =  0.7200000000000001D 02 +  0.1040000000000000D 03 I


NUMBER OF INITIAL APPROXIMATIONS GIVEN.   0
MAXIMUM NUMBER OF ITERATIONS.           200
TEST FOR CONVERGENCE.             0.10D-09
TEST FOR MULTIPLICITIES.          0.10D-01
RADIUS TO START SEARCH.           0.00D 00
RADIUS TO END SEARCH.             0.00D 00


BEFORE THE ATTEMPT TO IMPROVE ACCURACY, THE ZEROS OF P(X) ARE


 ROOTS OF P(X)                                            MULTIPLICITIES                INITIAL APPROXIMATION


  RDOT( 1) =  0.9999998836125019D 00 +  0.2000000052138284D 01 I        2         0.4829629115656279D 00 +  0.1294095284438187D 00 I
  RJDT( 2) =  0.1996737810257486D 01 +  0.1995253821143684D 01 I        3         0.7071067553046346D 00 +  0.7071068070684595D 00 I
  RDDT( 3) = -0.9902131979974624D 00 +  0.5142384322923812D 00 I-       1         SOLVED BY DIRECT METHOD

IN THE ATTEMPT TO IMPROVE ACCURACY, ROOT( 2) =  0.1996737810257486D 01 +  0.1995253821143684D 0I I DIO NOT CONVERGE.
THE PRESENT APPROXIMATION AFTER 200 ITERATIONS IS PRINTED BELOW.


AFTER  THE ATTEMPT TO IMPROVE ACCURACY, THE ZEROS OF P(X) ARE


 ROOTS OF P(X)                                            MULTIPLICITIES                INITIAL APPROXIMATION


  RODT( 1) =  0.9999998836125019D 00 +  0.2000000052138284D 01 I        2         0.4829629115656279D 00 +  0.1294095284438187D 00 I
  RODT( 2) =  0.1999992907503309D 01 +  0.1999959474001689D 01 I        3         0.7071067553046346D 00 +  0.7071068070684595D 00 I
  ROOT( 3) = -0.9999999999999998D 00 +  0.5000000000000000D 00 I        1         SOLVED BY DIRECT METHOD
```

Exhibit F. Roots Are: 2 + 2i (3), 1 + 2i (2), -1 + .5i (1).

```
      NEWTONS METHOD TO FIND ZEROS OF POLYNOMIALS
      POLYNOMIAL NUMBER 46 OF DEGREE  8


THE COEFFICIENTS OF P(X) ARE

  P(  1) =  0.1000000000000000D 01 +  0.0000000000000000D 00 I
  P(  2) =  0.0000000000000000D 00 + -0.3000000000000000D 01 I
  P(  3) = -0.1200000000000000D 02 + -0.2000000000000000D 01 I
  P(  4) =  0.4000000000000000D 01 +  0.2800000000000000D 02 I
  P(  5) =  0.6000000000000001D 02 +  0.0000000000000000D 00 I
  P(  6) = -0.1600000000000000D 02 + -0.1160000000000000D 03 I
  P(  7) = -0.1280000000000000D 03 +  0.8000000000000000D 01 I
  P(  8) =  0.0000000000000000D D0 +  0.1440000000000000D 03 I
  P(  9) =  0.6400000000000001D 02 + -0.3200000000000000D 02 I


NUMBER OF INITIAL APPROXIMATIONS GIVEN.   0
MAXIMUM NUMBER OF ITERATIONS.           200
TEST FOR CONVERGENCE.               0.10D-04
TEST FOR MULTIPLICITIES.            0.10D-01
RADIUS TO START SEARCH.             0.00D 00
RADIUS TO END SEARCH.               0.00D 00


BEFORE THE ATTEMPT TO IMPROVE ACCURACY, THE ZEROS OF P(X) ARE


 ROOTS OF P(X)                                                    MULTIPLICITIES                INITIAL APPROXIMATION


  ROOT( 1) =  0.9999080132939515D 0D +  0.9999050266032582D 00 I        4         0.4829629115656279D 00 +  0.1294095284438187D 00 I
  ROOT( 2) = -0.1856970314887275D 01 +  0.2861249814184128D-01 I        1         0.7071067553046346D 00 +  0.7071068070684595D 00 I
  ROOT( 3) = -0.2048443782235651D 01 + -0.1319376429987634D 00 I        1         0.3882284792654056D 00 +  0.1448888763117193D 01 I
  ROOT( 4) =  0.1999808633077203D 01 + -0.9998575268225686D 00 I        1         SOLVED BY DIRECT METHOD
  ROOT( 5) = -0.2094026589130082D 01 +  0.1035625652664575D 00 I        1         SOLVED BY DIRECT METHOD


AFTER  THE ATTEMPT TO IMPROVE ACCURACY, THE ZEROS OF P(X) ARE


 ROOTS OF P(X)                                                    MULTIPLICITIES                INITIAL APPROXIMATION


  ROOT( 1) =  0.9998985119617400D 00 +  0.9999213371300168D 00 I        4         0.4829629115656279D 00 +  0.1294095284438187D 00 I
  ROOT( 2) = -0.1999961557953157D 01 +  0.7862362479129184D-05 I        1         0.7071067553046346D 00 +  0.7071068070684595D 00 I
  ROOT( 3) = -0.2000007986928054D 01 + -0.2879153403652409D-04 I        1         0.3882284792654056D 00 +  0.1448888763117193D 01 I
  ROOT( 4) =  0.2000000000000041D 01 + -0.9999999999999982D 00 I        1         SOLVED BY DIRECT METHOD
  ROOT( 5) = -0.2000017849400097D 01 +  0.2356567534440272D-04 I        1         SOLVED BY DIRECT METHOD
```

Exhibit G. Roots Are: 1 + i (4), -2 (3), 2 - i (1).

```
MULLERS METHOD FOR FINDING THE ZEROS OF A POLYNOMIAL
POLYNOMIAL NUMBER 25 OF DEGREE  8

THE COEFFICIENTS OF P(X) ARE

 P( 1) =  0.1000000000000000D 01 +  0.0000000000000000D 00 I
 P( 2) = -0.5000000000000001D 01 + -0.II50000000000000 02 I
 P( 3) = -0.5175000000000001D 02 +  0.4300000000000001D 02 I
 P( 4) =  0.1572500000000000D 03 +  0.1446250000000000 03 I
 P( 5) =  0.3075000000000000D 03 + -0.3475000000000000D 03 I
 P( 6) = -0.4952500000000000D 03 + -0.4948750000000000 03 I
 P( 7) = -0.5857500000000001D 03 +  0.4247500000000001D 03 I
 P( 8) =  0.1810000000000000D 03 +  0.4420000000000001D 03 I
 P( 9) =  0.1580000000000000D 03 +  0.6000000000000001D 01 I


NUMBER OF INITIAL APPROXIMATIONS GIVEN.   0
MAXIMUM NUMBER OF ITERATIONS.           200
TEST FOR CONVERGENCE.                 0.10D-09
TEST FOR MULTIPLICITIES.              0.10D-01
RADIUS TO START SEARCH.               0.00D 00
RADIUS TO END SEARCH.                 0.00D 00


BEFORE ATTEMPT TO IMPROVE ACCURACY


ROOTS OF P(X)                                                   MULTIPLICITIES                INITIAL APPROXIMATION

 ROOT( 1) =  0.2000042202873018D 01 +  0.2000028743998294D 01 I       3          0.4829629115656279D 00 +  0.1294095284438187D 00 I
 ROOT( 2) =  0.1000563806908988D 01 +  0.1980638952568216D 01 I       1          0.7071067553046346D 00 +  0.7071068070684595D 00 I
 ROOT( 3) = -0.1063528418749844D 01 +  0.4942372379199120D 00 I       1          0.3882284792654056D 00 +  0.1448888763117193D 01 I
 ROOT( 4) = -0.9722122516869824D 00 +  0.5603797457973125D 00 I       1         -0.5176382551966724D 00 +  0.1931851608368755D 01 I
 ROOT( 5) =  0.9994370680506410D 00 +  0.2019094286928692D 01 I       1          SOLVED BY DIRECT METHOD
 ROOT( 6) = -0.9643868131418565D 00 +  0.4455635447909865D 00 I       1          SOLVED BY DIRECT METHOD


AFTER THE ATTEMPT TO IMPROVE ACCURACY


ROOTS OF P(X)                                                   MULTIPLICITIES                INITIAL APPROXIMATION

 ROOT( 1) =  0.2000042175889591D 01 +  0.2000028750713895D 01 I       3          0.4829629115656279D 00 +  0.1294095284438187D 00 I
 ROOT( 2) =  0.1000000023165255D 01 +  0.1999999805777700D 01 I       1          0.7071067553046346D 00 +  0.7071068070684595D 00 I
 ROOT( 3) = -0.1000006594122314D 01 +  0.4999952821126052D 00 I       1          0.3882284792654056D 00 +  0.1448888763117193D 01 I
 ROOT( 4) = -0.1000009725718865D 01 +  0.5000084175726651D 00 I       1         -0.5176382551966724D 00 +  0.1931851608368755D 01 I
 ROOT( 5) =  0.9999999168409400D 00 +  0.2000000100132659D 01 I       1          SOLVED BY DIRECT METHOD
 ROOT( 6) = -0.9999925738129557D 00 +  0.4999962818337703D 00 I       1          SOLVED BY DIRECT METHOD
```

Exhibit H. Roots Are: 2 + 2i (3), 1 + 2i (2), -1 + .5i (3).

```
 MULLERS METHOD FOR FINDING THE ZEROS OF A POLYNOMIAL
 POLYNOMIAL NUMBER 46 OF DEGREE  8

 THE COEFFICIENTS OF P(X) ARE

  P( 1) =  0.1000000E 01 +  0.0000000E 00 I
  P( 2) =  0.0000000E 00 + -0.3000000E 01 I
  P( 3) = -0.1200000E 02 + -0.2000000E 01 I
  P( 4) =  0.4000000E 01 +  0.2800000E 02 I
  P( 5) =  0.6000000E 02 +  0.0000000E 00 I
  P( 6) = -0.1600000E 02 + -0.1160000E 03 I
  P( 7) = -0.1280000E 03 +  0.8000000E 01 I
  P( 8) =  0.0000000E 00 +  0.1440000E 03 I
  P( 9) =  0.6400000E 02 + -0.3200000E 02 I


NUMBER OF INITIAL APPROXIMATIONS GIVEN.   0
MAXIMUM NUMBER OF ITERATIONS.           200
TEST FOR CONVERGENCE.              0.10E-04
TEST FOR MULTIPLICITIES.           0.10E-01
RADIUS TO START SEARCH.            0.00E 00
RADIUS TO END SEARCH.              0.00E 00


BEFORE ATTEMPT TO IMPROVE ACCURACY


ROOTS OF P(X)                                    MULTIPLICITIES          INITIAL APPROXIMATION

 ROOT( 1) =  0.1023019E 01 +  0.1016063E 01 I          2            0.4829629E 00 +  0.1294095E 00 I
 ROOT( 2) =  0.1000158E 01 +  0.9504023E 00 I          1            0.7071067E 00 +  0.7071068E 00 I
 ROOT( 3) =  0.9538472E 00 +  0.1017488E 01 I          1            0.3882295E 00 +  0.1448888E 01 I
 ROOT( 4) = -0.1973325E 01 + -0.2449018E-01 I          1           -0.5176364E 00 +  0.1931851E 01 I
 ROOT( 5) = -0.2033645E 01 + -0.1098356E-01 I          1           -0.1767765E 01 +  0.1767768E 01 I
 ROOT( 6) =  0.1999992E 01 + -0.9999957E 00 I          1            SOLVED BY DIRECT METHOD
 ROOT( 7) = -0.1993066E 01 +  0.3545329E-01 I          1            SOLVED BY DIRECT METHOD


AFTER THE ATTEMPT TO IMPROVE ACCURACY


ROOTS OF P(X)                                    MULTIPLICITIES          INITIAL APPROXIMATION

 ROOT( 1) =  0.1023855E 01 +  0.1017007E 01 I          2            0.4829629E 00 +  0.1294095E 00 I
 ROOT( 2) =  0.1004420E 01 +  0.9757368E 00 I          1            0.7071067E 00 +  0.7071068E 00 I
 ROOT( 3) =  0.9693815E 00 +  0.1017069E 01 I          1            0.3882295E 00 +  0.1448888E 01 I
 ROOT( 4) = -0.1988549E 01 + -0.1451340E-02 I          1           -0.5176364E 00 +  0.1931851E 01 I
 ROOT( 5) = -0.2006490E 01 + -0.9744499E-02 I          1           -0.1767765E 01 +  0.1767768E 01 I
 ROOT( 6) =  0.2000000E 01 + -0.1000000E 01 I          1            SOLVED BY DIRECT METHOD
 ROOT( 7) = -0.2004362E 01 +  0.1135974E-01 I          1            SOLVED BY DIRECT METHOD
```

Exhibit I. Roots Are: $1 + i$ (4), -2 (3), $2 - i$ (1).

```
 MULLERS METHOD FOR FINDING THE ZEROS OF A POLYNOMIAL
 POLYNOMIAL NUMBER 46 OF DEGREE  8


 THE COEFFICIENTS OF P(X) ARE


  P( 1) =  0.1000000000000000D 01 +  0.0000000000000000D 00 I
  P( 2) =  0.0000000000000000D 00 + -0.3000000000000000D 01 I
  P( 3) = -0.1200000000000000D 02 + -0.2000000000000000D 01 I
  P( 4) =  0.4000000000000000D 01 +  0.2800000000000000D 02 I
  P( 5) =  0.6000000000000001D 02 +  0.0000000000000000D 00 I
  P( 6) = -0.1600000000000000D 02 + -0.1160000000000000D 03 I
  P( 7) = -0.1280000000000000D 03 +  0.8000000000000000D 01 1
  P( 8) =  0.0000000000000000D 00 +  0.1440000000000000D 03 I
  P( 9) =  0.6400000000000001D 02 + -0.3200000000000000D 02 I


NUMBER OF INITIAL APPROXIMATIONS GIVEN.   0
MAXIMUM NUMBER OF ITERATIONS.            200
TEST FOR CONVERGENCE.               0.10D-04
TEST FOR MULTIPLICITIES.            0.10D-01
RADIUS TO START SEARCH.             0.00D 00
RADIUS TO END SEARCH.               0.00D 00


BEFORE ATTEMPT TO IMPROVE ACCURACY


ROOTS OF P(X)                                                 MULTIPLICITIES                INITIAL APPROXIMATION


 ROOT( 1) =  0.9999385361441768D 00 +  0.9999014841207898D 00 I          3           0.4829629115656279D 00 +  0.1294095284438187D 00 I
 ROOT( 2) =  0.1000184367790518D 01 +  0.1000295472219798D 01 I          1           0.7071067553046346D 00 +  0.7071068070684595D 00 I
 ROOT( 3) = -0.1995508869249228D 01 +  0.2597879064994983D-02 I          2           0.3882284792654056D 00 +  0.1448888763117193D 01 I
 ROOT( 4) =  0.1999994617928609D 01 + -0.1000018753040607D 01 I          1           SOLVED BY DIRECT METHOD
 ROOT( 5) = -0.2008976855653202D 01 + -0.5176929671550160D-02 I          1           SOLVED BY DIRECT METHOD


AFTER THE ATTEMPT TO IMPROVE ACCURACY


ROOTS OF P(X)                                                 MULTIPLICITIES                INITIAL APPROXIMATION


 ROOT( 1) =  0.1000179194957145D 01 +  0.1000267071195324D 01 I          3           0.4829629115656279D 00 +  0.1294095284438187D 00 I
 ROOT( 2) =  0.1000152145562312D 01 +  0.1000254437829650D 01 1          1           0.7071067553046346D 00 +  0.7071068070684595D 00 I
 ROOT( 3) = -0.1999984369889545D 01 + -0.3932859075256354D-04 I          2           0.3882284792654056D 00 +  0.1448888763117193D 01 I
 ROOT( 4) =  0.2000000000000000D 01 + -0.9999999999999998D 00 I          1           SOLVED BY DIRECT METHOD
 ROOT( 5) = -0.2000036150881542D 01 +  0.3663652712672905D-05 I          1           SOLVED BY DIRECT METHOD
```

Exhibit J. Roots Are: 1 + i (4), -2 (3), 2 - i (1).

```
          GREATEST COMMON DIVISOR METHOD USED WITH NEWTONS METHOD TO FIND ZEROS OF POLYNOMIALS
          POLYNOMIAL NUMBER 25 OF DEGREE  8


THE COEFFICIENTS OF P(X) ARE


  P( 1) =  0.1000000E 01 +  0.0000000E 00 I
  P( 2) = -0.5000000E 01 + -0.1150000E 02 I
  P( 3) = -0.5175000E 02 +  0.4300000E 02 I
  P( 4) =  0.1572500E 03 +  0.1446250E 03 I
  P( 5) =  0.3075000E 03 + -0.3475000E 03 I
  P( 6) = -0.4952500E 03 + -0.4948750E 03 I
  P( 7) = -0.5857500E 03 +  0.4247500E 03 I
  P( 8) =  0.1810000E 03 +  0.4420000E 03 I
  P( 9) =  0.1580000E 03 +  0.6000000E 01 I


NUMBER OF INITIAL APPROXIMATIONS GIVEN.    0
MAXIMUM NUMBER OF ITERATIONS.            200
TEST FOR ZERO IN SUBROUTINE GCD     0.10E-01
TEST FOR CONVERGENCE.               0.10E-04
TEST FOR MULTIPLICITIES.            0.10E 00
RADIUS TO START SEARCH.             0.00E 00
RADIUS TO END SEARCH.               0.00E 00


THE COEFFICIENTS OF H(X) = P(X)/G.C.D. ARE


  H( 1) =  0.1000000E 01 +  0.0000000E 00 I
  H( 2) = -0.2000005E 01 + -0.4499880E 01 I
  H( 3) = -0.7000046E 01 +  0.3500016E 01 I
  H( 4) =  0.9995270E 00 +  0.6999649E 01 I


BEFORE THE ATTEMPT TO IMPROVE ACCURACY, THE ZEROS OF P(X) ARE


ZEROS OF P(X)                                      MULTIPLICITIES          INITIAL APPROXIMATION

  ROOT( 1) =  0.9994811E 00 +  0.1999989E 01 I           1            0.4829629E 00 +  0.1294095E 00 I
  ROOT( 2) =  0.2000473E 01 +  0.1999909E 01 I           1           SOLVED BY DIRECT METHOD
  ROOT( 3) = -0.9999509E 00 +  0.4999819E 00 I           1           SOLVED BY DIRECT METHOD


AFTER  THE ATTEMPT TO IMPROVE ACCURACY, THE ZEROS OF P(X) ARE


ROOTS OF P(X)                                     MULTIPLICITIES          INITIAL APPROXIMATION

 ROOT( 1) =  0.9994814E 00 +  0.1999988E 01 I           2           0.4829629E 00 +  0.1294095E 00 I
 ROOT( 2) =  0.2000475E 01 +  0.1999910E 01 I           3           SOLVED BY DIRECT METHOD
 ROOT( 3) = -0.9999515E 00 +  0.4999818E 00 I           3           SOLVED BY DIRECT METHOD
```

Exhibit K. Roots Are: 2 + 2i (3), 1 + 2i (2), -1 + .5i (3).

```
          GREATEST COMMON DIVISOR METHOD USED WITH NEWTONS METHOD TO FIND ZEROS OF POLYNOMIALS
          POLYNOMIAL NUMBER 25 OF DEGREE  8


THE COEFFICIENTS OF P(X) ARE


  P( 1) =  0.1000000000000000D 01 +  0.0000000000000000D 00 I
  P( 2) = -0.5000000000000001D 01 + -0.1150000000000000D 02 I
  P( 3) = -0.5175000000000001D 02 +  0.4300000000000001D 02 I
  P( 4) =  0.1572500000000000D 03 +  0.1446250000000000D 03 I
  P( 5) =  0.3075000000000000D 03 + -0.3475000000000000D 03 I
  P( 6) = -0.4952500000000000D 03 + -0.4948750000000000D 03 I
  P( 7) = -0.5857500000000001D 03 +  0.4247500000000001D 03 I
  P( 8) =  0.1810000000000000D 03 +  0.4420000000000001D 03 I
  P( 9) =  0.1580000000000000D 03 +  0.6000000000000001D 01 I


NUMBER OF INITIAL APPROXIMATIONS GIVEN.   0
MAXIMUM NUMBER OF ITERATIONS.           200
TEST FOR ZERO IN SUBROUTINE GCD     0.10D-02
TEST FOR CONVERGENCE.               0.10D-09
TEST FOR MULTIPLICITIES.            0.10D-01
RADIUS TO START SEARCH.             0.00D 00
RADIUS TO END SEARCH.               0.00D 00


THE COEFFICIENTS OF H(X) = P(X)/G.C.D. ARE


  H( 1) =  0.1000000000000000D 01 +  0.0000000000000000D 00 I
  H( 2) = -0.2000000000000273D 01 + -0.4500000000000275D 01 I
  H( 3) = -0.7000000000000935D 01 +  0.3500000000000472D 01 I
  H( 4) =  0.9999999999993925D 00 +  0.7000000000000508D 01 I


BEFORE THE ATTEMPT TO IMPROVE ACCURACY, THE ZEROS OF P(X) ARE


 ROOTS OF P(X)                                                 MULTIPLICITIES               INITIAL APPROXIMATION


  ROOT( 1) =  0.9999999999995414D 00 +  0.1999999999999767D 01 I        1        0.4829629115656279D 00 +  0.1294095284438187D 00 I
  ROOT( 2) =  0.2000000000000671D 01 +  0.2000000000000530D 01 I        1        SOLVED BY DIRECT METHOD
  ROOT( 3) = -0.9999999999999402D 00 +  0.4999999999999780D 00 I        1        SOLVED BY DIRECT METHOD


AFTER  THE ATTEMPT TO IMPROVE ACCURACY, THE ZEROS OF P(X) ARE


ROOTS OF P(X)                                                  MULTIPLICITIES              INITIAL APPROXIMATION


  ROOT( 1) =  0.9999999999995416D 00 +  0.1999999999999767D 01 I        2        0.4829629115656279D 00 +  0.1294095284438187D 00 I
  ROOT( 2) =  0.2000000000000671D 01 +  0.2000000000000531D 01 I        3        SOLVED BY DIRECT METHOD
  ROOT( 3) = -0.9999999999999403D 00 +  0.4999999999999780D 00 I        3        SOLVED BY DIRECT METHOD
```

Exhibit L. Roots Are: 2 + 2i (3), 1 + 2i (2), -1 + .5i (3).

```
        GREATEST COMMON DIVISOR METHOD USED WITH NEWTONS METHOD TO FIND ZEROS OF POLYNOMIALS
        POLYNOMIAL NUMBER  9 OF DEGREE 12


THE COEFFICIENTS OF P(X) ARE


   P( 1) =  0.1000000000000000D 01 +  0.0000000000000000D 00 I
   P( 2) = -0.1200000000000000D 02 + -0.0000000000000000D 00 I
   P( 3) =  0.7200000000000001D 02 +  0.0000000000000000D 00 I
   P( 4) = -0.2800000000000000D 03 + -0.0000000000000000 00 I
   P( 5) =  0.7800000000000000D 03 +  0.0000000000000000D 00 I
   P( 6) = -0.1632000000000000D 04 + -0.0000000000000000 00 I
   P( 7) =  0.2624000000000000D 04 +  0.0000000000000000D 00 I
   P( 8) = -0.3264000000000000D 04 + -0.0000000000000000D 00 I
   P( 9) =  0.3120000000000000D 04 +  0.0000000000000000D 00 I
   P(10) = -0.2240000000000000D 04 + -0.0000000000000000D 00 I
   P(11) =  0.1152000000000000D 04 +  0.0000000000000000D 00 I
   P(12) = -0.3840000000000000 03 + -0.0000000000000000D 00 I
   P(13) =  0.6400000000000001D 02 +  0.0000000000000000 00 I


NUMBER OF INITIAL APPROXIMATIONS GIVEN.   0
MAXIMUM NUMBER OF ITERATIONS.           200
TEST FOR ZERO IN SUBROUTINE GCD    0.10D-02
TEST FOR CONVERGENCE.              0.10D-09
TEST FOR MULTIPLICITIES.           0.10D-01
RADIUS TO START SEARCH.            0.00D 00
RADIUS TO END SEARCH.              0.00D 00


THE COEFFICIENTS OF H(X) = P(X)/G.C.D. ARE


   H( 1) =  0.1000000000000000D 01 +  0.0000000000000000D 00 I
   H( 2) = -0.2000000000000015D 01 + -0.0000000000000000D 00 I
   H( 3) =  0.1999999999999830D 01 +  0.0000000000000000D 00 I


ZEROS OF P(X)                                                MULTIPLICITIES


   ROOT( 1) =  0.1000000000000007D 01 +  0.9999999999999074D 00 I         6          SOLVED BY DIRECT METHOD
   ROOT( 2) =  0.1000000000000007D 01 + -0.9999999999999074D 00 I         6          SOLVED BY DIRECT METHOD
```

Exhibit M. Roots Are: 1 + i (6), 1 - i (6).

```
          GREATEST COMMON DIVISOR METHOD USED WITH MULLERS METHOD TO FIND ZEROS OF POLYNOMIALS
          POLYNOMIAL NUMBER 12 OF DEGREE 15


THE COEFFICIENTS OF P(X) ARE


  P( 1) =  0.4800000000000000D 02 +  0.0000000000000000D 00 I
  P( 2) =  0.2557120000000000D 03 + -0.3840000000000000D 03 I
  P( 3) = -0.7353556800000000D 02 + -0.2189696000000000D 04 I
  P( 4) = -0.3855565696000000D 04 + -0.6946851456000001D 04 I
  P( 5) = -0.1733386464800000D 05 + -0.1420625972800000D 05 I
  P( 6) = -0.4967989270400001D 05 + -0.1765857464000000D 05 I
  P( 7) = -0.1022394572130000D 06 + -0.6030664232000001D 04 I
  P( 8) = -0.1642742200560000D 06 +  0.4137366230400000D 05 I
  P( 9) = -0.2036625888420000D 06 +  0.1093899227670000D 06 I
  P(10) = -0.1871255780010000D 06 +  0.1929865440330000D 06 I
  P(11) = -0.1274997298590000D 06 +  0.2171341227420000D 06 I
  P(12) = -0.2814692716800000D 05 +  0.1928489727960000D 06 I
  P(13) =  0.1329434434800000D 05 +  0.1038130226550000D 06 I
  P(14) =  0.3053900774700000D 05 +  0.2998989141300000D 05 I
  P(15) = -0.1835899020000000D 03 + -0.1827632160000000D 03 I
  P(16) =  0.2755620000000000D 00 +  0.2755620000000000D 00 I


NUMBER OF INITIAL APPROXIMATIONS GIVEN.   0
MAXIMUM NUMBER OF ITERATIONS.           200
TEST FOR ZERO IN SUBROUTINE GCD    0.10D-02
TEST FOR CONVERGENCE.              0.10D-09
TEST FOR MULTIPLICITIES.           0.10D-01
RADIUS TO START SEARCH.            0.00D 00
RADIUS TO END SEARCH.              0.00D 00


THE COEFFICIENTS OF H(X) = P(X)/G.C.D. ARE


  H( 1) =  0.4800000000000000D 02 +  0.0000000000000000D 00 I
  H( 2) =  0.1598559999959083D 03 + -0.1440000000018463D 03 I
  H( 3) =  0.2675199999866287D 03 + -0.5275680000068238D 03 I
  H( 4) =  0.3271959999781904D 03 + -0.9144160000075942D 03 I
  H( 5) =  0.2301599999604451D 02 + -0.1521252000003768D 04 I
  H( 6) = -0.7207200004799557D 02 + -0.1327427999996661D 04 I
  H( 7) = -0.7557840000541411D 03 + -0.7520040000333157D 03 I
  H( 8) =  0.2268000133545684D 01 +  0.2267999925897489D 01 I


BEFORE ATTEMPT TO IMPROVE ACCURACY


ROOTS OF P(X)                                                 MULTIPLICITIES                INITIAL APPROXIMATION


 ROOT( 1) =  0.3000000038688574D-02 + -0.1370060304840657D-09 I         1          0.4829629115656279D 00 +  0.1294095284438187D 00 I
 ROOT( 2) =  0.1465369392888612D-09 +  0.1000000000302531D 01 I         1          0.7071067553046346D 00 +  0.7071068070694595D 00 I
```

Exhibit N.

```
 ROOT( 3) = -0.1625381061484938D-09 +  0.1499999999807065D 01 I         1          0.3882284792654056D 00 +  0.1448888763117193D 01 I
 ROOT( 4) =  0.1140410054459155D-09 +  0.3000000000053946D 01 I         1         -0.5176382551966724D 00 +  0.1931851608368755D 01 I
 ROOT( 5) = -0.2333333333340885D 01 +  0.4287264124730616D-11 I         1         -0.1767767147080701D 01 +  0.1767766758852015D 01 I
 ROOT( 6) = -0.1167273685117228D-10 + -0.1500000000006768D 01 I         1          SOLVED BY DIRECT METHOD
 ROOT( 7) = -0.1000000000032260D 01 + -0.9999999999855917D 00 I         1          SOLVED BY DIRECT METHOD


IN THE ATTEMPT TO IMPROVE ACCURACY, ROOT( 1) =  0.3000000038688574D-02 + -0.1370060304840657D-09 1
DID NOT CONVERGE AFTER 200 ITERATIONS
THE PRESENT APPROXIMATION IS  0.3003000000103452D-02 + -0.1371430347506515D-09 I



AFTER THE ATTEMPT TO IMPROVE ACCURACY


ROOTS OF P(X)                                                  MULTIPLICITIES               INITIAL APPROXIMATION


 ROOT( 1) =  0.1465369084191118D-09 +  0.1000000000302531D 01 I         2          0.7071067553046346D 00 +  0.7071068070684595D 00 I
 ROOT( 2) = -0.1625379402673067D-09 +  0.1499999999807065D 01 I         2          0.3882284792654056D 00 +  0.1448888763117193D 01 I
 ROOT( 3) =  0.1140410804253790D-09 +  0.3000000000053946D 01 I         3         -0.5176382551966724D 00 +  0.1931851608368755D 01 I
 ROOT( 4) = -0.2333333333340885D 01 +  0.4287665075729792D-11 I         1         -0.1767767147080701D 01 +  0.1767766758852015D 01 I
 ROOT( 5) = -0.1167210227268680D-10 + -0.1500000000006767D 01 I         2          SOLVED BY DIRECT METHOD
 ROOT( 6) = -0.1000000000032260D 01 + -0.9999999999855942D 00 I         3          SOLVED BY DIRECT METHOD
COMPILE TIME=    18.25 SEC,EXECUTION TIME=    11.78 SEC,OBJECT CODE=   35720 BYTES,ARRAY AREA=   10440 BYTES,UNUSED=   23840 BYTES

COMPILE TIME=     0.09 SEC,EXECUTION TIME=     0.00 SEC,OBJECT CODE=   35720 BYTES,ARRAY AREA=   10440 BYTES,UNUSED=   23840 BYTES
        $STOP
```

Exhibit N. Roots Are: -2.33 (1), .003 (2), i (2),
1.5i (2), -1.5i (2), 3i (3), -1 - i (3).

```
        NEWTONS METHOD TO FIND ZEROS OF POLYNOMIALS
        POLYNOMIAL NUMBER 25 OF DEGREE  8


THE COEFFICIENTS OF P(X) ARE

   P( 1) =  0.10000000000000000D 01 +  0.00000000000000000D 00 I
   P( 2) = -0.50000000000000001D 01 + -0.11500000000000000D 02 I
   P( 3) = -0.51750000000000001D 02 +  0.43000000000000001D 02 I
   P( 4) =  0.15725000000000000D 03 +  0.14462500000000000D 03 I
   P( 5) =  0.30750000000000000D 03 + -0.34750000000000000 03 I
   P( 6) = -0.49525000000000000D 03 + -0.49487500000000000D 03 I
   P( 7) = -0.58575000000000001D 03 +  0.42475000000000001D 03 I
   P( 8) =  0.18100000000000000D 03 +  0.44200000000000001D 03 I
   P( 9) =  0.15800000000000000 03 +  0.60000000000000001D 01 I


NUMBER OF INITIAL APPROXIMATIONS GIVEN.    0
MAXIMUM NUMBER OF ITERATIONS.            200
TEST FOR CONVERGENCE.               0.10D-04
TEST FOR MULTIPLICITIES.            0.10D-01
RADIUS TO START SEARCH.             0.00D 00
RADIUS TO END SEARCH.               0.00D 00


BEFORE THE ATTEMPT TO IMPROVE ACCURACY, THE ZEROS OF P(X) ARE


 ROOTS OF P(X)                                                   MULTIPLICITIES                 INITIAL APPROXIMATION


  ROOT( 1) =  0.99999395980181380 00 +  0.19999876973381040 01 I        2         0.48296291156562790 00 +  0.12940952844381870 00 I
  ROOT( 2) = -0.98425374559631380 00 +  0.50549074650384760 00 I        1         0.70710675530463460 00 +  0.70710680706845950 00 I
  ROOT( 3) =  0.20009317781427920 01 +  0.19774929769036070 01 I        2         0.38822847926540560 00 +  0.14488887631171930 01 I
  ROOT( 4) = -0.10107747992624520 01 +  0.53828182384544560 00 I        1        -0.51763825519667240 00 +  0.19318516083687550 01 I
  ROOT( 5) =  0.19989623336444980 01 +  0.20446779366249860 01 I        1         SOLVED BY DIRECT METHOD
  ROOT( 6) = -0.10057852646749430 01 +  0.45658814454229950 00 I        1         SOLVED BY DIRECT METHOD


AFTER  THE ATTEMPT TO IMPROVE ACCURACY, THE ZEROS OF P(X) ARE


 ROOTS OF P(X)                                                   MULTIPLICITIES                 INITIAL APPROXIMATION


  ROOT( 1) =  0.99999698109077980 00 +  0.19999938490897910 01 I        2         0.48296291156562790 00 +  0.12940952844381870 00 I
  ROOT( 2) = -0.99998441512125260 00 +  0.50000531974058520 00 I        1         0.70710675530463460 00 +  0.70710680706845950 00 I
  ROOT( 3) =  0.20000003156277770 01 +  0.19999403726276780 01 I        2         0.38822847926540560 00 +  0.14488887631171930 01 I
  ROOT( 4) = -0.10000042124759280 01 +  0.50001764461645580 00 I        1        -0.51763825519667240 00 +  0.19318516083687550 01 I
  ROOT( 5) =  0.19999441804788620 01 +  0.20000119831885490 01 I        1         SOLVED BY DIRECT METHOD
  ROOT( 6) = -0.10000022649123540 01 +  0.49997996483755230 00 I        1         SOLVED BY DIRECT METHOD
```

Exhibit O. Roots Are: 2 + 2i (3), 1 + 2i (2), -1 + .5i (3).

```
        NEWTONS METHOD TO FIND ZEROS OF POLYNOMIALS
        POLYNOMIAL NUMBER 27 OF DEGREE  4


THE COEFFICIENTS OF P(X) ARE

  P( 1) =  0.10000000000000000D 01 +  0.00000000000000000D 00 I
  P( 2) = -0.41000000000000000D 00 + -0.41000000000000000D 00 I
  P( 3) = -0.00000000000000000D 00 +  0.12607000000000000D 00 I
  P( 4) =  0.86140999999999998D-02 + -0.86140999999999998D-02 I
  P( 5) = -0.44142009600000000D-03 + -0.00000000000000000D 00 I


NUMBER OF INITIAL APPROXIMATIONS GIVEN.   0
MAXIMUM NUMBER OF ITERATIONS.           200
TEST FOR CONVERGENCE.               0.10D-09
TEST FOR MULTIPLICITIES.            0.10D-01
RADIUS TO START SEARCH.             0.00D 00
RADIUS TO END SEARCH.               0.00D 00


BEFORE THE ATTEMPT TO IMPROVE ACCURACY, THE ZEROS OF P(X) ARE


 ROOTS OF P(X)                                                     MULTIPLICITIES                INITIAL APPROXIMATION


  ROOT( 1) =  0.10199999996271430 00 +  0.10199999995993400 00 I         1          0.48296291156562790 00 +  0.12940952844381870 00 I
  ROOT( 2) =  0.10399999998699890 00 +  0.10399999998594790 00 I         2          0.70710675530463460 00 +  0.70710680706845950 00 I
  ROOT( 3) =  0.10000000006328800 00 +  0.10000000006817020 00 I         1          SOLVED BY DIRECT METHOD


AFTER  THE ATTEMPT TO IMPROVE ACCURACY, THE ZEROS OF P(X) ARE


 ROOTS OF P(X)                                                     MULTIPLICITIES                INITIAL APPROXIMATION


  ROOT( 1) =  0.10199999997118460 00 +  0.10199999995146360 00 I         1          0.48296291156562790 00 +  0.12940952844381870 00 I
  ROOT( 2) =  0.10399999998728120 00 +  0.10399999998566560 00 I         2          0.70710675530463460 00 +  0.70710680706845950 00 I
  ROOT( 3) =  0.10100000000315340 00 +  0.10099999999679770 00 I         1          SOLVED BY DIRECT METHOD
```

Exhibit P. Roots Are: .101 + .101i, .102 + .102i, .103 + .103i, .104 + .104i.

```
        NEWTONS METHOD TO FIND ZEROS OF POLYNOMIALS
        POLYNOMIAL NUMBER 27 OF DEGREE  4


THE COEFFICIENTS OF P(X) ARE

   P( 1) =  0.10000000000000000D 01 +  0.00000000000000000 00 I
   P( 2) = -0.41000000000000000D 00 + -0.41000000000000000D 00 I
   P( 3) = -0.00000000000000000D 00 +  0.12607000000000000D 00 I
   P( 4) =  0.86140999999999998D-02 + -0.86140999999999998D-02 I
   P( 5) = -0.44142009600000000D-03 + -0.00000000000000000D 00 I


NUMBER OF INITIAL APPROXIMATIONS GIVEN.    0
MAXIMUM NUMBER OF ITERATIONS.            200
TEST FOR CONVERGENCE.                0.10D-09
TEST FOR MULTIPLICITIES.             0.10D-17
RADIUS TO START SEARCH.              0.00D 00
RADIUS TO END SEARCH.                0.00D 00


BEFORE THE ATTEMPT TO IMPROVE ACCURACY, THE ZEROS OF P(X) ARE


 ROOTS OF P(X)                                                MULTIPLICITIES                INITIAL APPROXIMATION


   ROOT( 1) =  0.1019999999627143D 00 +  0.1019999999599340D 00 I        1          0.4829629115656279D 00 +  0.1294095284438187D 00 I
   ROOT( 2) =  0.1039999999869989D 00 +  0.1039999999859479D 00 I        1          0.7071067553046346D 00 +  0.7071068070684595D 00 I
   ROOT( 3) =  0.1030000000381370D 00 +  0.1030000000411039D 00 I        1          SOLVED BY DIRECT METHOD
   ROOT( 4) =  0.1010000000121498D 00 +  0.1010000000130142D 00 I        1          SOLVED BY DIRECT METHOD


AFTER  THE ATTEMPT TO IMPROVE ACCURACY, THE ZEROS OF P(X) ARE


 ROOTS OF P(X)                                                MULTIPLICITIES                INITIAL APPROXIMATION


   ROOT( 1) =  0.1019999999711846D 00 +  0.1019999999514636D 00 I        1          0.4829629115656279D 00 +  0.1294095284438187D 00 I
   ROOT( 2) =  0.1039999999872812D 00 +  0.1039999999856656D 00 I        1          0.7071067553046346D 00 +  0.7071068070684595D 00 I
   ROOT( 3) =  0.1030000000182317D 00 +  0.1030000000339041D 00 I        1          SOLVED BY DIRECT METHOD
   ROOT( 4) =  0.1009999999969032D 00 +  0.1009999999966382D 00 I        1          SOLVED BY DIRECT METHOD
```

Exhibit Q. Roots Are: .101 + .101i, .102 + .102i, .103 + .103i, .104 + .104i.

```
 MULLERS METHOD FOR FINOING THE ZEROS OF A POLYNOMIAL
 POLYNOMIAL NUMBER 27 OF OEGREE  4


 THE COEFFICIENTS OF P(X) ARE


  P( 1) =  0.1000000000000000D 01 +  0.0000000000000000 00 I
  P( 2) = -0.4100000000000000D 00 + -0.4100000000000000D 00 I
  P( 3) = -0.0000000000000000D 00 +  0.1260700000000000D 00 I
  P( 4) =  0.8614099999999998D-02 + -0.8614099999999998D-02 I
  P( 5) = -0.4414200960000000-03 + -0.0000000000000000 00 I


NUMBER OF INITIAL APPROXIMATIONS GIVEN.   0
MAXIMUM NUMBER OF ITERATIONS.           200
TEST FOR CONVERGENCE.                 0.10D-09
TEST FOR MULTIPLICITIES.              0.10D-17
RADIUS TO START SEARCH.               0.000 00
RADIUS TO ENO SEARCH.                 0.00D 00


BEFORE ATTEMPT TO IMPROVE ACCURACY


ROOTS OF P(X)                                                      MULTIPLICITIES                INITIAL APPROXIMATION


 ROOT( 1) =  0.1039999999869266D 00 +  0.1039999999874569D 00 I          1          0.4829629115656279D 00 +  0.1294095284438187D 00 I
 ROOT( 2) =  0.1030000000382713D 00 +  0.1030000000366408D 00 I          1          0.7071067553046346D 00 +  0.7071068070684595D 00 I
 ROOT( 3) =  0.1019999999626628D 00 +  0.1019999999643326D 00 I          1          SOLVED BY DIRECT METHOD
 ROOT( 4) =  0.1010000000121392D 00 +  0.1010000000115695D 00 I          1          SOLVED BY DIRECT METHOD


AFTER THE ATTEMPT TO IMPROVE ACCURACY


ROOTS OF P(X)                                                      MULTIPLICITIES                INITIAL APPROXIMATION


 ROOT( 1) =  0.1039999999869295D 00 +  0.1039999999874673D 00 I          1          0.4829629115656279D 00 +  0.1294095284438187D 00 I
 ROOT( 2) =  0.1030000000244983D 00 +  0.1030000000151221D 00 I          1          0.7071067553046346D 00 +  0.7071068070684595D 00 I
 ROOT( 3) =  0.1019999999829168D 00 +  0.1019999999717330D 00 I          1          SOLVED BY DIRECT METHOD
 ROOT( 4) =  0.1010000000055776D 00 +  0.1010000000055703D 00 I          1          SOLVED BY DIRECT METHOD
```

Exhibit R. Roots Are: .101 + .101i, .102 + .102i, .103 + .103i, .104 + .104i.

```
MULLERS METHOD FOR FINDING THE ZEROS OF A POLYNOMIAL
POLYNOMIAL NUMBER 33 OF DEGREE  4


THE COEFFICIENTS OF P(X) ARE

 P( 1) =  0.1000000000000000D 01 +  0.0000000000000000D 00 I
 P( 2) = -0.4046000000000000D 01 + -0.4086000000000000D 01 I
 P( 3) = -0.1219800000000000D 00 +  0.1239896200000000D 02 I
 P( 4) =  0.8525941492000000D 01 + -0.8277958252000000D 01 I
 P( 5) = -0.4269975877160000D 01 + -0.8402356471999999D-01 I


NUMBER OF INITIAL APPROXIMATIONS GIVEN.   0
MAXIMUM NUMBER OF ITERATIONS.           200
TEST FOR CONVERGENCE.             0.10D-09
TEST FOR MULTIPLICITIES.          0.10D-17
RADIUS TO START SEARCH.           0.00D 00
RADIUS TO END SEARCH.             0.00D 00


BEFORE ATTEMPT TO IMPROVE ACCURACY


ROOTS OF P(X)                                                 MULTIPLICITIES               INITIAL APPROXIMATION

 ROOT( 1) =  0.1010000013641422D 01 +  0.1019999997158714D 01 I        1         0.4829629115656279D 00 +  0.1294095284438187D 00 I
 ROOT( 2) =  0.1010999959008221D 01 +  0.1021000085163440 01 I         1         0.7071067553046346D 00 +  0.7071068070684595D 00 I
 ROOT( 3) =  0.1012999986290457D 01 +  0.1023000002835164D 01 I        1         SOLVED BY DIRECT METHOD
 ROOT( 4) =  0.1012000041059900D 01 +  0.1021999991489779D 01 I        1         SOLVED BY DIRECT METHOD


AFTER THE ATTEMPT TO IMPROVE ACCURACY


ROOTS OF P(X)                                                 MULTIPLICITIES               INITIAL APPROXIMATION

 ROOT( 1) =  0.1010999939030615D 01 +  0.1021000048991936D 01 I        1         0.4829629115656279D 00 +  0.1294095284438187D 00 I
 ROOT( 2) =  0.1011999992454635D 01 +  0.1022000010646505D 01 I        1         0.7071067553046346D 00 +  0.7071068070684595D 00 I
 ROOT( 3) =  0.1012999994777002D 01 +  0.1022999994537829D 01 I        1         SOLVED BY DIRECT METHOD
 ROOT( 4) =  0.1012000096743678D 01 +  0.1022000043617003D 01 I        1         SOLVED BY DIRECT METHOD
```

Exhibit S. Roots Are: 1.010 + 1.020i, 1.011 + 1.021i,
1.012 + 1.022i, 1.013 + 1.023i.

```
          GREATEST COMMON DIVISOR METHOD USED WITH NEWTONS METHOD TO FIND ZEROS OF POLYNOMIALS
          POLYNOMIAL NUMBER 27 OF DEGREE  4


THE COEFFICIENTS OF P(X) ARE


  P( 1) =  0.10000000000000000D 01 +  0.00000000000000000D 00 I
  P( 2) = -0.41000000000000000D 00 + -0.41000000000000000D 00 I
  P( 3) = -0.00000000000000000D 00 +  0.12607000000000000D 00 I
  P( 4) =  0.86140999999999998D-02 + -0.86140999999999998D-02 I
  P( 5) = -0.44142009600000000D-03 + -0.00000000000000000D 00 I


NUMBER OF INITIAL APPROXIMATIONS GIVEN.   0
MAXIMUM NUMBER OF ITERATIONS.           200
TEST FOR ZERO IN SUBROUTINE GCD    0.10D-02
TEST FOR CONVERGENCE.              0.10D-09
TEST FOR MULTIPLICITIES.           0.10D-01
RADIUS TO START SEARCH.            0.00D 00
RADIUS TO END SEARCH.              0.00D 00


THE COEFFICIENTS OF H(X) = P(X)/G.C.D. ARE


  H( 1) =  0.10000000000000000D 01 +  0.00000000000000000D 00 I
  H( 2) = -0.10250000000000000D 00 + -0.10250000000000000D 00 I


ZEROS OF P(X)                                               MULTIPLICITIES


  ROOT( 1) =  0.10250000000000000D 00 +  0.10250000000000000D 00 I       4       SOLVED BY DIRECT METHOD
```

Exhibit T. Roots Are: .101 + .101i, .102 + .102i, .103 + .103i, .104 + .104i.

```
          GREATEST COMMON DIVISOR METHOD USED WITH NEWTONS METHOD TO FIND ZEROS OF POLYNOMIALS
          POLYNOMIAL NUMBER 27 OF DEGREE  4


THE COEFFICIENTS OF P(X) ARE


   P( 1) =  0.1000000000000000D 01 +  0.0000000000000000D 00 I
   P( 2) = -0.4100000000000000D 00 + -0.4100000000000000D 00 I
   P( 3) = -0.0000000000000000D 00 +  0.1260700000000000D 00 I
   P( 4) =  0.8614099999999998D-02 + -0.8614099999999998D-02 I
   P( 5) = -0.4414200960000000D-03 + -0.0000000000000000D 00 I


NUMBER OF INITIAL APPROXIMATIONS GIVEN.    0
MAXIMUM NUMBER OF ITERATIONS.            200
TEST FOR ZERO IN SUBROUTINE GCD     0.10D-19
TEST FOR CONVERGENCE.               0.10D-09
TEST FOR MULTIPLICITIES.            0.10D-09
RADIUS TO START SEARCH.             0.00D 00
RADIUS TO END SEARCH.               0.00D 00


THE COEFFICIENTS OF H(X) = P(X)/G.C.D. ARE


   H( 1) =  0.1000000000000000D 01 +  0.0000000000000000D 00 I
   H( 2) = -0.4100000000000000D 00 + -0.4100000000000000D 00 I
   H( 3) = -0.0000000000000000D 00 +  0.1260700000000000D 00 I
   H( 4) =  0.8614099999999998D-02 + -0.8614099999999998D-02 I
   H( 5) = -0.4414200960000000D-03 + -0.0000000000000000D 00 I


BEFORE THE ATTEMPT TO IMPROVE ACCURACY, THE ZEROS OF P(X) ARE


 ROOTS OF P(X)                                                   MULTIPLICITIES                INITIAL APPROXIMATION


   ROOT( 1) =  0.1019999999627143D 00 +  0.1019999999599340D 00 I        1        0.4829629115656279D 00 +  0.1294095284438187D 00 I
   ROOT( 2) =  0.1039999999869989D 00 +  0.1039999999859479D 00 I        1        0.7071067553046346D 00 +  0.7071068070684595D 00 I
   ROOT( 3) =  0.1030000000381370D 00 +  0.1030000000411039D 00 I        1        SOLVED BY DIRECT METHOD
   ROOT( 4) =  0.1010000000121498D 00 +  0.1010000000130142D 00 I        1        SOLVED BY DIRECT METHOD


AFTER  THE ATTEMPT TO IMPROVE ACCURACY, THE ZEROS OF P(X) ARE


ROOTS OF P(X)                                                   MULTIPLICITIES                INITIAL APPROXIMATION


   ROOT( 1) =  0.1019999999711846D 00 +  0.1019999999514636D 00 I        1        0.4829629115656279D 00 +  0.1294095284438187D 00 I
   ROOT( 2) =  0.1039999999872812D 00 +  0.1039999999856656D 00 I        1        0.7071067553046346D 00 +  0.7071068070684595D 00 I
   ROOT( 3) =  0.1030000000182317D 00 +  0.1030000000339041D 00 I        1        SOLVED BY DIRECT METHOD
   ROOT( 4) =  0.1009999999969032D 00 +  0.1009999999966382D 00 I        1        SOLVED BY DIRECT METHOD
```

Exhibit U. Roots Are: .101 + .101i, .102 + .102i, .103 + .103i, .104 + .104i.

```
          NEWTONS METHOD TO FIND ZEROS OF POLYNOMIALS
          POLYNOMIAL NUMBER 33 OF DEGREE  4


THE COEFFICIENTS OF P(X) ARE

   P( 1) =  0.1000000000000000D 01 +  0.0000000000000000 00 I
   P( 2) = -0.4046000000000000 01 + -0.4086000000000000 01 I
   P( 3) = -0.1219800000000000D 00 +  0.1239896200000000 02 I
   P( 4) =  0.8525941492000000D 01 + -0.8277958252000000 01 I
   P( 5) = -0.4269975877160000D 01 + -0.8402356471999999D-01 I


NUMBER OF INITIAL APPROXIMATIONS GIVEN.   0
MAXIMUM NUMBER OF ITERATIONS.           200
TEST FOR CONVERGENCE.             0.10D-07
TEST FOR MULTIPLICITIES.          0.10D-06
RADIUS TO START SEARCH.           0.00D 00
RADIUS TO END SEARCH.             0.00D 00


BEFORE THE ATTEMPT TO IMPROVE ACCURACY, THE ZEROS OF P(X) ARE


 ROOTS OF P(X)                                                MULTIPLICITIES              INITIAL APPROXIMATION


  ROOT( 1) =  0.1010000020242410D 01 +  0.1020000000258103D 01 I        1          0.4829629115656279 00 +  0.1294095284438187 00 I
  ROOT( 2) =  0.1010999939199952D 01 +  0.1020999999178984D 01 I        1          0.7071067553046346D 00 +  0.7071068070684595D 00 I
  ROOT( 3) =  0.1012999979682521D 01 +  0.102299999969727 4 0 01 I        1          SOLVED BY DIRECT METHOD
  ROOT( 4) =  0.1012000060875117D 01 +  0.1022000000865640D 01 I        1          SOLVED BY DIRECT METHOD


AFTER  THE ATTEMPT TO IMPROVE ACCURACY, THE ZEROS OF P(X) ARE


 ROOTS OF P(X)                                                MULTIPLICITIES              INITIAL APPROXIMATION


  ROOT( 1) =  0.1010000013301666D 01 +  0.1020000007195040 01 I         I          0.4829629115656279D 00 +  0.1294095284438187 00 I
  ROOT( 2) =  0.1010999960033530D 01 +  0.1020999978369735D 01 I         1          0.7071067553046346D 00 +  0.7071068070684595D 00 I
  ROOT( 3) =  0.1012999986621423D 01 +  0.1022999992757863D 01 I         1          SOLVED BY DIRECT METHOD
  ROOT( 4) =  0.1012000040062344D 01 +  0.1021999966196982D 01 I         1          SOLVED BY DIRECT METHOD
```

Exhibit V. Roots Are: 1.010 + 1.020i, 1.011 + 1.021i, 1.012 + 1.022i, 1.013 + 1.023i.

```
      NEWTONS METHOD TO FIND ZEROS OF POLYNOMIALS
      POLYNOMIAL NUMBER 37 OF DEGREE  6


THE COEFFICIENTS OF P(X) ARE

   P( 1) =  0.1000000000000000D 01 +  0.0000000000000000D 00 I
   P( 2) = -0.1104600000000000D 02 + -0.7086000000000001D 01 I
   P( 3) =  0.3794202000000000D 02 +  0.6713896200000001D 02 I
   P( 4) =  0.1476868749200000D 02 + -0.2412407522520000D 03 I
   P( 5) = -0.2650544690771600D 03 +  0.3033533037232799D 03 I
   P( 6) =  0.3330998887979599D 03 + -0.4935380807147997D 02 I
   P( 7) = -0.9276313939143995D 02 + -0.6162618070408000D 02 I


NUMBER OF INITIAL APPROXIMATIONS GIVEN.   0
MAXIMUM NUMBER OF ITERATIONS.           200
TEST FOR CONVERGENCE.               0.10D-09
TEST FOR MULTIPLICITIES.            0.10D-17
RADIUS TO START SEARCH.             0.00D 00
RADIUS TO END SEARCH.               0.00D 00


NO CONVERGENCE FOR THE FOLLOWING APPROXIMATIONS AFTER 200 ITERATIONS.


 0.4829629115656279D 00 +  0.1294095284438187D 00 I          INITIAL APPROXIMATION
-0.4829629115656279D 00 + -0.1294095284438187D 00 I          ALTERED APPROXIMATION


BEFORE THE ATTEMPT TO IMPROVE ACCURACY, THE ZEROS OF P(X) ARE


 ROOTS OF P(X)                                                MULTIPLICITIES          INITIAL APPROXIMATION


  ROOT( 1) =  0.1009999396981447D 01 +  0.1019999951561502D 01 I       1         0.1294094930884686D 00 +  0.4829629210390644D 00 I
  ROOT( 2) =  0.1011001815251888D 01 +  0.1021000145224417D 01 I       1         0.7071067553046346D 00 +  0.7071068070684595D 00 I
  ROOT( 3) =  0.1011998180643081D 01 +  0.1021999853333628D 01 I       1         0.3882284792654056D 00 +  0.1448888763117193D 01 I
  ROOT( 4) =  0.1013000607113586D 01 +  0.1023000049880457D 01 I       1        -0.5176382551966724D 00 +  0.1931851608368755D 01 I
  ROOT( 5) =  0.3999999999999999D 01 + -0.2000000000000001D 01 I       1         SOLVED BY DIRECT METHOD
  ROOT( 6) =  0.3000000000000001D 01 +  0.4999999999999998D 01 I       1         SOLVED BY DIRECT METHOD


AFTER  THE ATTEMPT TO IMPROVE ACCURACY, THE ZEROS OF P(X) ARE


 ROOTS OF P(X)                                                MULTIPLICITIES          INITIAL APPROXIMATION


  ROOT( 1) =  0.1009999396981447D 01 +  0.1019999951561502D 01 I       1         0.1294094930884686D 00 +  0.4829629210390644D 00 I
  ROOT( 2) =  0.1011001873747942D 01 +  0.1021000398198728D 01 I       1         0.7071067553046346D 00 +  0.7071068070684595D 00 I
  ROOT( 3) =  0.1011998062406218D 01 +  0.1021999572143372D 01 I       1         0.3882284792654056D 00 +  0.1448888763117193D 01 I
  ROOT( 4) =  0.1013000626481528D 01 +  0.1023000853439890D 01 I       1        -0.5176382551966724D 00 +  0.1931851608368755D 01 I
  ROOT( 5) =  0.4000000000000001D 01 + -0.2000000000000001D 01 I       1         SOLVED BY DIRECT METHOD
  ROOT( 6) =  0.3000000000000000D 01 +  0.5000000000000001D 01 I       1         SOLVED BY DIRECT METHOD
```

Exhibit W. Roots Are: 1.010 + 1.020i, 1.011 + 1.021i, 1.012 + 1.022i, 1.013 + 1.023i, 4 - 2i, 3 + 5i.

```
          GREATEST COMMON DIVISOR METHOD USED WITH NEWTONS METHOD TO FIND ZEROS OF POLYNOMIALS
          POLYNOMIAL NUMBER 55 OF DEGREE  9


THE COEFFICIENTS OF P(X) ARE


   P( 1) =  0.1000000000000000D 01 +  0.0000000000000000D 00 I
   P( 2) =  0.1320230000000000D 02 + -0.2403000000000000D 00 I
   P( 3) =  0.5562569530000001D 02 + -0.1214820634000000D 02 I
   P( 4) =  0.4450453890000001D 02 + -0.1268725824200000D 03 I
   P( 5) = -0.3230724061600000D 03 + -0.4865587937200000D 03 I
   P( 6) = -0.1349266779280000D 04 + -0.6694374018400001D 03 I
   P( 7) = -0.2678658554260000D 04 + -0.1648687451800000D 03 I
   P( 8) = -0.2299189598500000D 04 +  0.2622280045000000D 03 I
   P( 9) = -0.4124338680000000D 03 +  0.4266134510000001D 03 I
   P(10) =  0.1281640000000000D 01 +  0.4358727000000001D 02 I


NUMBER OF INITIAL APPROXIMATIONS GIVEN.    0
MAXIMUM NUMBER OF ITERATIONS.            200
TEST FOR ZERO IN SUBROUTINE GCD    0.10D-02
TEST FOR CONVERGENCE.              0.10D-09
TEST FOR MULTIPLICITIES.           0.10D-01
RADIUS TO START SEARCH.            0.00D 00
RADIUS TO END SEARCH.              0.00D 00


THE COEFFICIENTS OF H(X) = P(X)/G.C.D. ARE


   H( 1) =  0.1000000000000000D 01 +  0.0000000000000000D 00 I
   H( 2) =  0.4101149994288120D 01 + -0.2120149996474013D 01 I
   H( 3) = -0.9835700015798032D 01 + -0.1368289996947017D 02 I
   H( 4) = -0.3757344988763747D 02 + -0.5113449941508815D 01 I
   H( 5) = -0.5414099972072969D 02 +  0.5369949985430600D 02 I
   H( 6) =  0.9500001341688850D 00 +  0.1106499937561766D 02 I


BEFORE THE ATTEMPT TO IMPROVE ACCURACY, THE ZEROS OF P(X) ARE


 ROOTS OF P(X)                                               MULTIPLICITIES              INITIAL APPROXIMATION


  ROOT( 1) = -0.1011499923894657D 00 +  0.1201499951350621D 00 I        1         0.4829629115656279D 00 +  0.1294095284438187D 00 I
  ROOT( 2) = -0.1488614390879477D-08 +  0.2000000000230853D 01 I        1         0.7071067553046346D 00 +  0.7071068070684595D 00 I
  ROOT( 3) = -0.2000000000264861D 01 + -0.9999999989340373D 00 I        1         0.3882284792654056D 00 +  0.1448888763117193D 01 I
  ROOT( 4) =  0.2999999999770455D 01 +  0.9999999999898534D 00 I        1         SOLVED BY DIRECT METHOD
  ROOT( 5) = -0.4999999999915633D 01 +  0.5228084631880847D-10 I        1         SOLVED BY DIRECT METHOD


AFTER  THE ATTEMPT TO IMPROVE ACCURACY, THE ZEROS OF P(X) ARE


ROOTS OF P(X)                                               MULTIPLICITIES              INITIAL APPROXIMATION


  ROOT( 1) = -0.1011499923894657D 00 +  0.1201499951350621D 00 I        2         0.4829629115656279D 00 +  0.1294095284438187D 00 I
  ROOT( 2) = -0.1488614375951514D-08 +  0.2000000000230853D 01 I        1         0.7071067553046346D 00 +  0.7071068070684595D 00 I
  ROOT( 3) = -0.2000000000264861D 01 + -0.9999999989340374D 00 I        3         0.3882284792654056D 00 +  0.1448888763117193D 01 I
  ROOT( 4) =  0.2999999999770455D 01 +  0.9999999999898546D 00 I        1         SOLVED BY DIRECT METHOD
  ROOT( 5) = -0.4999999999915634D 01 +  0.5228039001585336D-10 I        2         SOLVED BY DIRECT METHOD
```

Exhibit X. Roots Are: −2 − i (3), −5 (2), 3 + i, 2i,
−0.1011 + .1201i, −.1012 + .1202i.

```
 MULLERS METHOD FOR FINDING THE ZEROS OF A POLYNOMIAL
 POLYNOMIAL NUMBER 55 OF DEGREE  9

 THE COEFFICIENTS OF P(X) ARE

  P( 1) =  0.1000000000000000D 01 +  0.0000000000000000D 00 I
  P( 2) =  0.1320230000000000D 02 + -0.2403000000000000D 00 I
  P( 3) =  0.5562569530000001D 02 + -0.1214820634000000D 02 I
  P( 4) =  0.4450453890000001D 02 + -0.1268725824200000D 03 I
  P( 5) = -0.3230724061600000D 03 + -0.4865587937200000D 03 I
  P( 6) = -0.1348266779280000D 04 + -0.6694374018400001D 03 I
  P( 7) = -0.2678658554260000D 04 + -0.1648687451800000D 03 I
  P( 8) = -0.2299188598500000D 04 +  0.2622280045000000D 03 I
  P( 9) = -0.4124338680000000D 03 +  0.4266134510000001D 03 I
  P(10) =  0.1281640000000000D 01 +  0.4358727000000001D 02 I


NUMBER OF INITIAL APPROXIMATIONS GIVEN.   9
MAXIMUM NUMBER OF ITERATIONS.           200
TEST FOR CONVERGENCE.             0.10D-09
TEST FOR MULTIPLICITIES.          0.10D-12
RADIUS TO START SEARCH.           0.00D 00
RADIUS TO END SEARCH.             0.00D 00


BEFORE ATTEMPT TO IMPROVE ACCURACY


ROOTS OF P(X)                                                   MULTIPLICITIES                INITIAL APPROXIMATION


 ROOT( 1) = -0.1011999999999818D 00 +  0.1202000000000116D 00 I         1         -0.1011499923894657D 00 +  0.1201499951350621D 00 I
 ROOT( 2) = -0.1011000000000182D 00 +  0.1200999999999884D 00 I         1         -0.1011499923894657D 00 +  0.1201499951350621D 00 I
 ROOT( 3) = -0.2754555594079078D-15 +  0.2000000000000000D 01 I         1         -0.1488614375951514D-08 +  0.2000000000230853D 01 I
 ROOT( 4) = -0.2000000124605528D 01 + -0.9999999638510141D 00 I         1         -0.2000000000264861D 01 + -0.9999999989340374D 00 I
 ROOT( 5) = -0.1999999953794536D 01 + -0.1000000161591884D 01 I         1         -0.2000000000264861D 01 + -0.9999999989340374D 00 I
 ROOT( 6) = -0.1999999921599928D 01 + -0.9999998745570992D 00 I         1         -0.2000000000264861D 01 + -0.9999999989340374D 00 I
 ROOT( 7) =  0.3000000000000000D 01 +  0.1000000000000000D 01 I         1          0.2999999999770455D 01 +  0.9999999999898546D 00 I
 ROOT( 8) = -0.4999999964575374D 01 +  0.1006811217499438D-06 I         1          SOLVED BY DIRECT METHOD
 ROOT( 9) = -0.5000000035424633D 01 + -0.1006811245532569D-06 I         1          SOLVED BY DIRECT METHOD


IN THE ATTEMPT TO IMPROVE ACCURACY, ROOT( 7) =  0.3000000000000000D 01 +  0.1000000000000000D 01 I
DID NOT CONVERGE AFTER 200 ITERATIONS
THE PRESENT APPROXIMATION IS  0.3002999961376190D 01 +  0.1000999987125397D 01 I



AFTER THE ATTEMPT TO IMPROVE ACCURACY


ROOTS OF P(X)                                                   MULTIPLICITIES                INITIAL APPROXIMATION


 ROOT( 1) = -0.1011999999999725D 00 +  0.1202000000000043D 00 I         1         -0.1011499923894657D 00 +  0.1201499951350621D 00 I
 ROOT( 2) = -0.1011999999999796D 00 +  0.1202000000000046D 00 I         1         -0.1011499923894657D 00 +  0.1201499951350621D 00 I
 ROOT( 3) = -0.1951126961795227D-15 +  0.2000000000000000D 01 I         1         -0.1488614375951514D-08 +  0.2000000000230853D 01 I
 ROOT( 4) = -0.2000000125319424D 01 + -0.9999999634117842D 00 I         1         -0.2000000000264861D 01 + -0.9999999989340374D 00 I
 ROOT( 5) = -0.1999999955943858D 01 + -0.1000000162709050D 01 I         1         -0.2000000000264861D 01 + -0.9999999989340374D 00 I
 ROOT( 6) = -0.2000000011069415D 01 + -0.9999998285019765D 00 I         1         -0.2000000000264861D 01 + -0.9999999989340374D 00 I
 ROOT( 7) = -0.4999999964166074D 01 +  0.1005693679791196D-06 I         1          SOLVED BY DIRECT METHOD
 ROOT( 8) = -0.5000000035616385D 01 + -0.1005712562663813D-06 I         1          SOLVED BY DIRECT METHOD
```

Exhibit Y. Roots Are: $-2 - i$ (3), -5 (2), $3 + i$, $2i$,
$-.1011 + .1201i$, $-.1012 + .1202i$.

```
MULLERS METHOD FOR FINDING THE ZEROS OF A POLYNOMIAL
POLYNOMIAL NUMBER 55 OF DEGREE  9


THE COEFFICIENTS OF P(X) ARE


 P( 1) =  0.1000000000000000D 01 +  0.0000000000000000D 00 I
 P( 2) =  0.1320230000000000D 02 + -0.2403000000000000D 00 I
 P( 3) =  0.5562569530000001D 02 + -0.1214820634000000D 02 I
 P( 4) =  0.4450453890000001D 02 + -0.1268725824200000D 03 I
 P( 5) = -0.3230724061600000D 03 + -0.4865587937200000D 03 I
 P( 6) = -0.1348266779280000D 04 + -0.6694374018400001D 03 I
 P( 7) = -0.2678658554260000D 04 + -0.1648687451800000D 03 I
 P( 8) = -0.2299188598500000D 04 +  0.2622280045000000D 03 I
 P( 9) = -0.4124338680000000D 03 +  0.4266134510000001D 03 I
 P(10) =  0.1281640000000000D 01 +  0.4358727000000001D 02 I


NUMBER OF INITIAL APPROXIMATIONS GIVEN.   0
MAXIMUM NUMBER OF ITERATIONS.           200
TEST FOR CONVERGENCE.              0.10D-09
TEST FOR MULTIPLICITIES.           0.10D-11
RADIUS TO START SEARCH.            0.00D 00
RADIUS TO END SEARCH.              0.00D 00


BEFORE ATTEMPT TO IMPROVE ACCURACY


ROOTS OF P(X)                                                MULTIPLICITIES                 INITIAL APPROXIMATION


 ROOT( 1) = -0.1011000000000203D 00 +  0.1201000000000003D 00 I        1           0.4829629115656279D 00 +  0.1294095284438187D 00 I
 ROOT( 2) = -0.1011999999999797D 00 +  0.1201999999999998D 00 I        1           0.7071067553046346D 00 +  0.7071068070684595D 00 I
 ROOT( 3) = -0.1995228819321680D-15 +  0.2000000000000000D 01 I        1           0.3882284792654056D 00 +  0.1448888763117193D 01 I
 ROOT( 4) = -0.2000007154743688D 01 + -0.9999828065437014D 00 I        1          -0.5176382551966724D 00 +  0.1931851608368755D 01 I
 ROOT( 5) = -0.4999999955556945D 01 +  0.1127307532752009D-06 I        1          -0.1767767147080701D 01 +  0.1767766758852015D 01 I
 ROOT( 6) = -0.1999981533202796D 01 + -0.1000002400471554D 01 I        1          -0.2897777583074990D 01 +  0.7764567463987070D 00 I
 ROOT( 7) = -0.2000011312053508D 01 + -0.1000014792984741D 01 I        1          -0.3380740248331229D 01 + -0.9058671940816160D 00 I
 ROOT( 8) =  0.3000000000000000D 01 +  0.9999999999999989D 00 I        1           SOLVED BY DIRECT METHOD
 ROOT( 9) = -0.5000000044443064D 01 + -0.1127307561743507D-06 I        1           SOLVED BY DIRECT METHOD


IN THE ATTEMPT TO IMPROVE ACCURACY, ROOT( 4) = -0.2000007154743688D 01 + -0.9999828065437014D 00 I
DID NOT CONVERGE AFTER 200 ITERATIONS
THE PRESENT APPROXIMATION IS -0.2000008660261324D 01 + -0.9999838754875147D 00 I



AFTER THE ATTEMPT TO IMPROVE ACCURACY


ROOTS OF P(X)                                                MULTIPLICITIES                 INITIAL APPROXIMATION


 ROOT( 1) = -0.1011999999999823D 00 +  0.1202000000000176D 00 I        1           0.4829629115656279D 00 +  0.1294095284438187D 00 I
 ROOT( 2) = -0.1011999999999802D 00 +  0.1201999999999995D 00 I        1           0.7071067553046346D 00 +  0.7071068070684595D 00 I
 ROOT( 3) = -0.2368948183084774D-15 +  0.2000000000000000D 01 I        1           0.3882284792654056D 00 +  0.1448888763117193D 01 I
 ROOT( 4) = -0.4999999956768864D 01 +  0.1129904903095989D-06 I        1          -0.1767767147080701D 01 +  0.1767766758852015D 01 I
 ROOT( 5) = -0.1999982781479067D 01 + -0.9999960391562971D 00 I        1          -0.2897777583074990D 01 +  0.7764567463987070D 00 I
 ROOT( 6) = -0.2000007036875357D 01 + -0.1000015580846282D 01 I        1          -0.3380740248331229D 01 + -0.9058671940816160D 00 I
 ROOT( 7) =  0.3000000000000000D 01 +  0.1000000000000000D 01 I        1           SOLVED BY DIRECT METHOD
 ROOT( 8) = -0.5000000044052519D 01 + -0.1131737422302264D-06 I        1           SOLVED BY DIRECT METHOD
```

Exhibit Z. Roots Are: $-2 - i$ (3), -5 (2), $3 + i$, $2i$,
$-.1011 + .1201i$, $-.1012 + .1202i$.

BIBLIOGRAPHY

1. Beckett, Royce and James Hurt. Numerical Calculations and Algorithms. New York: McGraw Hill, 1967.

2. Conte, S. D. Elementary Numerical Analysis. New York: McGraw Hill, 1965.

3. Henrici, Peter. Elements of Numerical Analysis. New York: John Wiley and Sons, Inc., 1964.

4. Lehmer, D. H. "A Machine Method for Solving Polynomial Equations." Jour. Assoc. Comp. Mach., Vol. 8 (1961), 151-162.

5. Muller, David E. "A Method for Solving Algebraic Equations Using an Automatic Computer." Math Tables and Aids to Comp., Vol. 10 (1956), 208-215.

6. McCalla, Thomas Richard. Introduction to Numerical Methods and Fortran Programming. New York: John Wiley and Sons, Inc., 1967.

7. Barnes, Wilfred E. Introduction to Abstract Algebra. Boston: D. C. Heath and Company, 1963.

8. Markushevich, A. I. Theory of Functions of a Complex Variable. Englewood Cliffs, N.J.: Printice-Hall, Inc., 1965.

9. Nehari, Zeev. Introduction to Complex Analysis. Boston: Allyn and Bacon, Inc., 1968.

10. Watkins, Bruce O. "Finding Zeros of a Polynomial by the Q-D Algorithm." Communications of the Assoc. of Comp. Mach., Vol. 8 (1965), 570-574.

11. Fröberg, Carl-Erik. Introduction to Numerical Analysis. Reading, Mass.: Addison-Wesley Publishing Co., Inc., 1965.

12. Henrici, Peter. "The Quotient-Difference Algorithm." Natl. Bureau of Standards Applied Mathematics Series, Vol. 49 (1958), 23-46.

APPENDIX A

SPECIAL FEATURES OF THE PROGRAMS

Several special features have been provided in each program as an aid to the user and to improve accuracy of the results. These are explained and illustrated below.[1]

1. Generating Approximations

If the user does not have initial approximations available, subroutine GENAPP can systematically generate, for an N^{th} degree polynomial, N initial approximations of increasing magnitude, beginning with the magnitude specified by XSTART. If XSTART is 0., XSTART is automatically initialized to 0.5 to avoid the approximation 0. + 0.i. The approximations are generated according to the formula:

$$X_K = (\text{XSTART} + 0.5K)\ (\text{Cos}\ \beta + i\ \text{Sin}\ \beta)$$

where

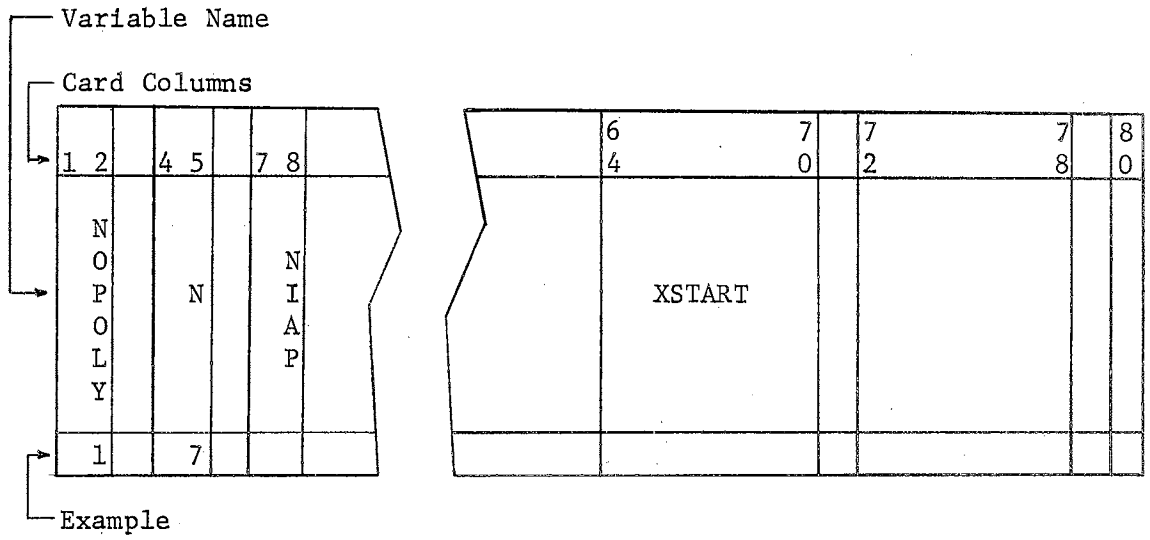
$$\beta = \frac{\Pi}{12} + K\frac{\Pi}{6}\ ,\quad K = 0,1,2,\ldots$$

To accomplish this, the user defined the number of initial approximations to be read (NIAP) on the control card to be zero (0) or these

[1]These illustrations are representative of Newton's method in double precision. Control cards for single precision and other methods should be prepared accordingly.

columns (7-8) may be left blank. If XSTART is left blank, it is interpreted as 0.

For example, a portion of a control card which generates initial approximations beginning at the origin for a seventh degree polynomial is shown in example 1.



Example 1

The approximations are generated in a spiral configuration as illustrated in figure 4. Exhibit B of Chapter VII is an example of output resulting from generated approximations.

Example 2 shows a portion of a control card which generated initial approximations beginning at a magnitude of 25.0 for a sixth degree polynomial.

1 2		4 5		7 8		64 70		72 78		80
NOPOLY		N		NIAP		XSTART				
1		6				2.5D + 01				

Example 2

Note that if the approximations are generated beginning at the origin, the order in which the roots are found will probably be of increasing magnitude. Roots obtained in this way are usually more accurate.

2. Altering Approximations

If an initial approximation, X_0, does not produce convergence to a root within the maximum number of iterations, it is systematically altered a maximum of five times until convergence is possibly obtained according to the following formulas:

If the number of the alteration is odd: $(j = 1,3)$

$$X_{j+1} = |X_0| \; (\text{Cos } \beta + i \text{ Sin } \beta) \text{ where}$$

$$\beta = \text{Tan}^{-1} \frac{\text{Im } X_0}{\text{Re } X_0} + K \frac{\Pi}{3} ; \quad K = 1 \text{ if } j = 1, \; 2 \text{ if } j = 3.$$

If the number of the alteration is even: $(j = 0,2,4)$

$$X_{j+1} = -X_j .$$

Each altered approximation is then taken as a starting approximation. Each initial or altered approximation which does not produce convergence is printed as in Exhibit AA. If none of the six starting approximations produce convergence, the next initial approximation is taken, and the process repeated. The six approximations are spaced 60 degrees apart on a circle of radius $|X_0|$ centered at the origin as illustrated in figure 5.

3. Searching the Complex Plane

By use of initial approximations and the altering technique, any region of the complex plane in the form of an annulus centered at the origin can be searched for roots. This procedure can be accomplished in two ways.

The first way is more versatile but requires more effort on the part of the user. Specifically selected initial approximations can be used to define particular regions to be searched. For example, if the roots of a particular polynomial are known to have magnitudes between 20 and 40, an annulus of inner radius 20 and outer radius 40 could be searched by using the initial approximations 20. + i, 23. + i, 26. + i, 29. + i, 32. + i, 35. + i, 38. + i, 40. + i.

By generating initial approximations internally, the program can search an annulus centered at the origin of inner radius XSTART and outer radius XEND. Values for XSTART and XEND are supplied on the control card by the user. Example 3 shows a portion of a control card to search the above annulus of inner radius 20.0 and outer radius 40.0.

1 2		4 5		7 8			64 70		72 78		80
NOPOLY		N		NIAP			XSTART		XEND		
1		7					2.0D + 01		4.0D + 01		

Example 3

Note that since not less than N initial approximations can be generated at one time, the outer radius of the annulus actually searched may be greater than XEND but not greater than XEND + .5N.

Example 4 shows a control card to search a circle of radius 15.

1 2		4 5		7 8			64 70		72 78		80
NOPOLY		N		NIAP			XSTART		XEND		
2		7							1.5D + 01		

Example 4

Figure 6 shows the distribution of initial and altered approximations for an annulus of width 2 and inner radius a.

4. Improving Zeros Found

After the zeros of a polynomial are found, they are printed under the heading "Before the Attempt to Improve Accuracy." They are then used as initial approximations with Newton's (Muller's) method applied each time to the full (undeflated) polynomial. In most cases, zeros that have lost accuracy due to roundoff error in the deflation process are improved. The improved zeros are then printed under the heading "After the Attempt to Improve Accuracy." Since each root is used as an approximation to the original (undeflated) polynomial, it is possible that the root may converge to an entirely different root. As an example, see exhibit S of Chapter VII. This is especially true where several zeros are close together. Therefore, the user should check both lists of zeros to determine whether or not this has occurred. An example of improved roots is given in exhibit G of Chapter VII.

5. Solving Quadratic Polynomial

After N-2 roots of an Nth degree polynomial have been extracted, the remaining quadratic, $aX^2 + bX + c$, is solved using the quadratic formula

$$X = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

for the two remaining roots. These are indicated by the words "Solved By Direct Method" in the initial approximation column. If only a polynomial of degree 1 is to be solved, the solution is found directly as $(X - C) = 0$ implies $X = C$.

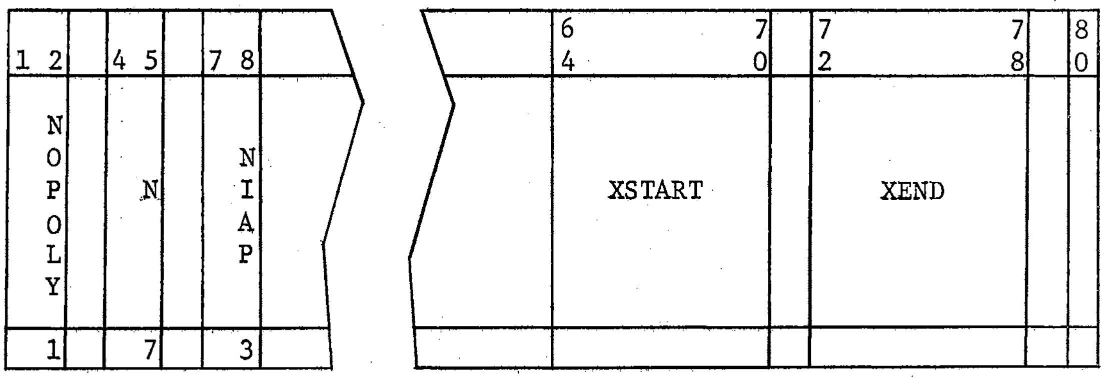
6. Missing Roots

If not all N roots of an Nth degree polynomial are found, the coefficients of the remaining deflated polynomial are printed under the heading "Coefficients of Deflated Polynomial For Which No Zeros Were Found." The user may then work with this polynomial in an attempt to find the remaining roots. The coefficient of the highest degree term will be printed first (exhibit BB).

7. Miscellaneous

By using various combinations of values for NIAP, XSTART, and XEND, the user has several options available as illustrated below.

Example 5 shows the control card for a seventh degree polynomial. Three initial approximations are supplied by the user. At most three distinct roots will be found and the remaining deflated polynomial will be printed (exhibit BB).



Example 5

Note that if several roots are known to the user, they may be "divided out" of the original polynomial by using this procedure.

Example 6 indicates that 2 initial approximations are supplied by the user to a 7th degree polynomial. After these approximations are used the circle or radius 15 will be searched for the remaining roots.

1 2		4 5		7 8		64 70		72 78		80
NOPOLY		N		NIAP		XSTART		XEND		
1		7		2				1.5D + 01		

Example 6

By defining XSTART between 0. and 15. an annulus instead of the circle will be searched (exhibit CC).

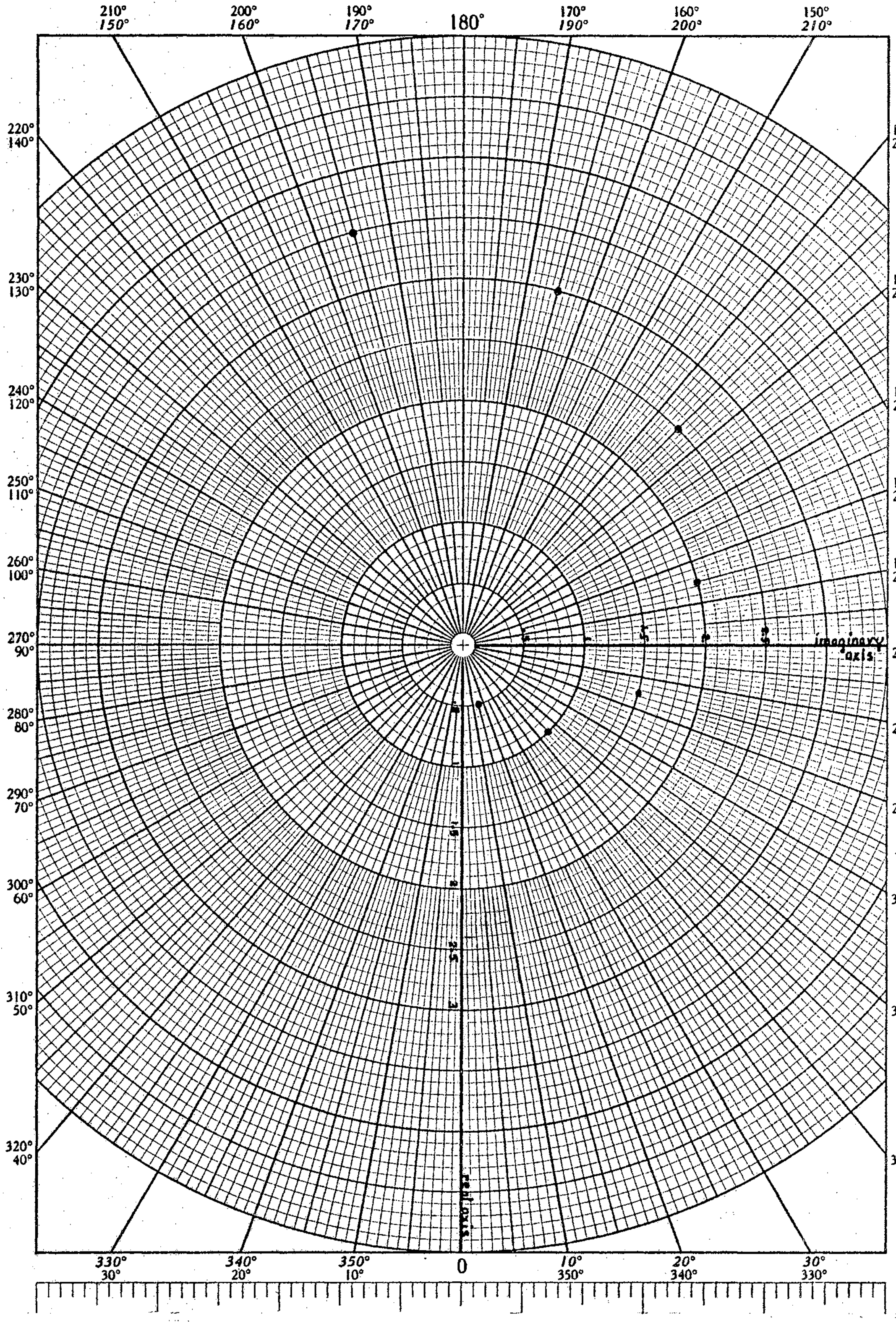


Figure 4. Generating Initial Approximations

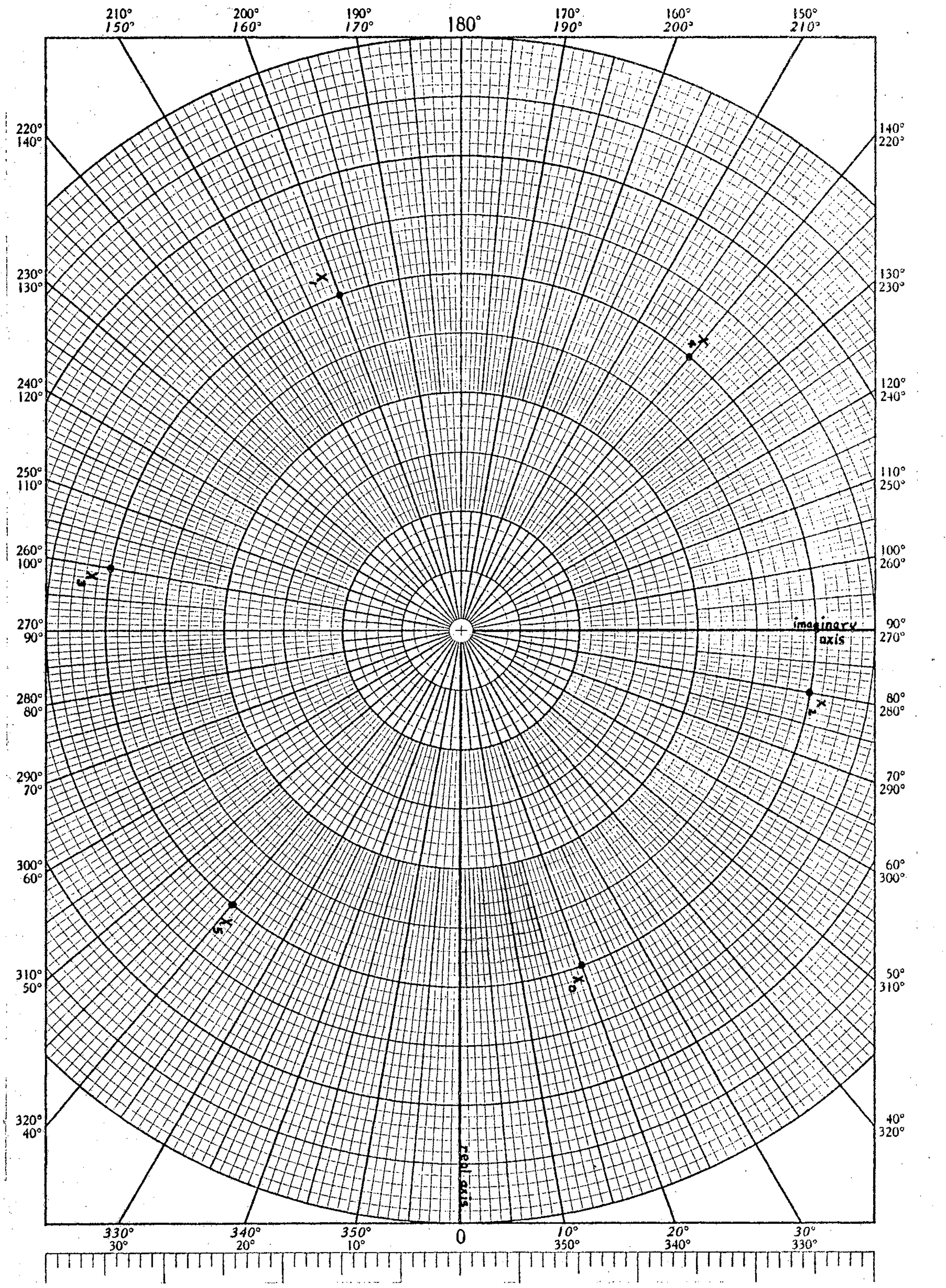


Figure 5. Altering Approximations

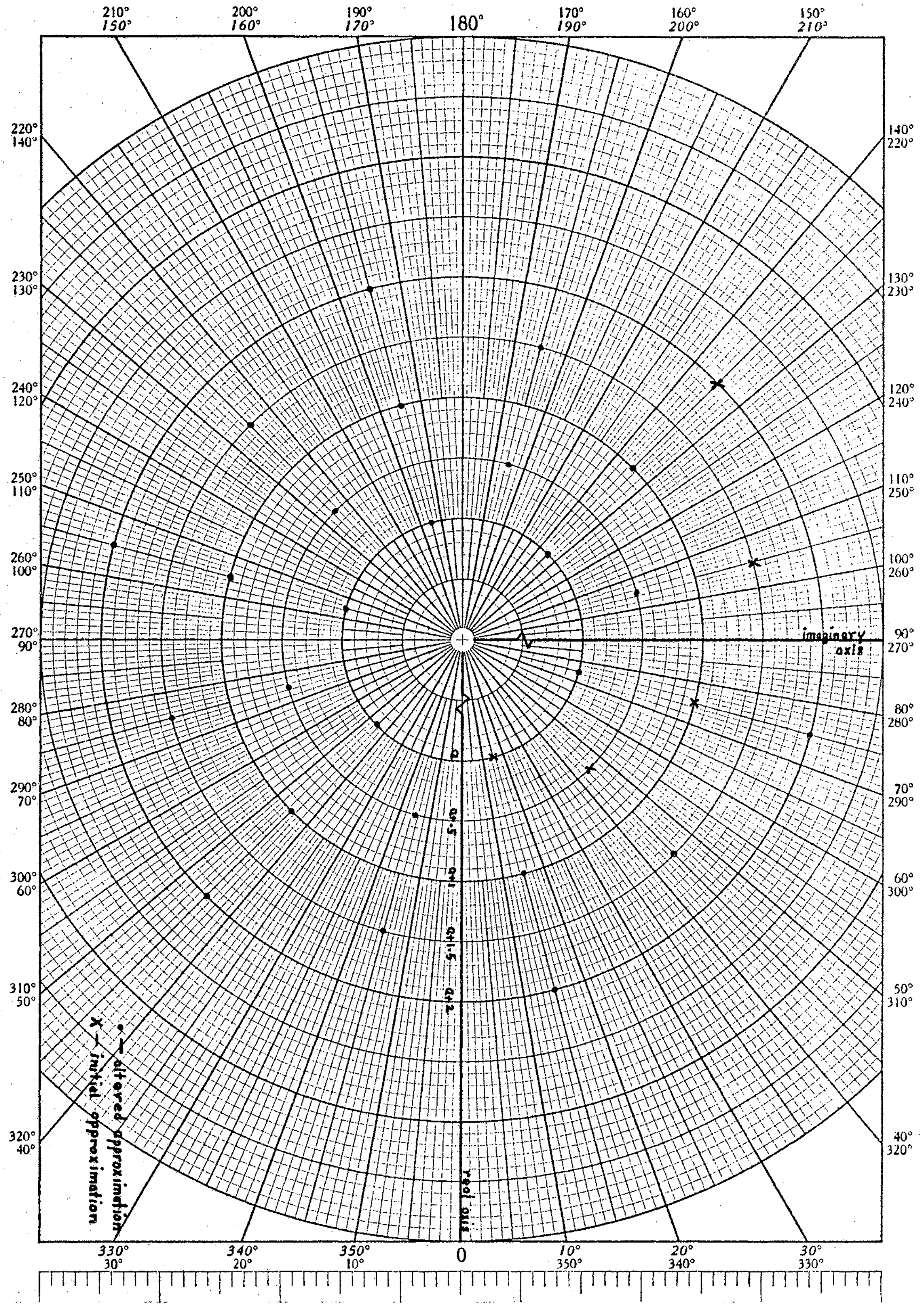


Figure 6. Distribution of Approximations

```
          NEWTONS METHOD TO FIND ZEROS OF POLYNOMIALS
          POLYNOMIAL NUMBER  2 OF DEGREE  3


THE COEFFICIENTS OF P(X) ARE

  P( 1) =  0.1000000000000000D 01 +  0.0000000000000000D 00 I
  P( 2) =  0.2000000000000000 01 +  0.0000000000000000D 00 I
  P( 3) = -0.1000000000000000D 01 + -0.0000000000000000D 00 I
  P( 4) = -0.2000000000000000D 01 + -0.0000000000000000D 00 I


NUMBER OF INITIAL APPROXIMATIONS GIVEN.    0
MAXIMUM NUMBER OF ITERATIONS.              3
TEST FOR CONVERGENCE.               0.10D-03
TEST FOR MULTIPLICITIES.            0.10D-01
RADIUS TO START SEARCH.             0.00D 00
RADIUS TO END SEARCH.               0.00D 00


NO CONVERGENCE FOR THE FOLLOWING APPROXIMATIONS AFTER   3 ITERATIONS.


 0.4829629115656279D 00 +  0.1294095284438187D 00 I          INITIAL APPROXIMATION
-0.4829629115656279D 00 + -0.1294095284438187D 00 I          ALTERED APPROXIMATION
 0.1294094930884686D 00 +  0.4829629210390644D 00 I          ALTERED APPROXIMATION
-0.1294094930884686D 00 + -0.4829629210390644D 00 I          ALTERED APPROXIMATION
-0.3535534294161402D 00 +  0.3535533517704030D 00 I          ALTERED APPROXIMATION
 0.3535534294161402D 00 + -0.3535533517704030D 00 I          ALTERED APPROXIMATION

 0.7071067553046346D 00 +  0.7071068070684595D 00 I          INITIAL APPROXIMATION
-0.7071067553046346D 00 + -0.7071068070684595D 00 I          ALTERED APPROXIMATION
-0.2588191275983359D 00 +  0.9659258041843774D 00 I          ALTERED APPROXIMATION
 0.2588191275983359D 00 + -0.9659258041843774D 00 I          ALTERED APPROXIMATION
-0.9659258610249968D 00 +  0.2588189154662357D 00 I          ALTERED APPROXIMATION
 0.9659258610249968D 00 + -0.2588189154662357D 00 I          ALTERED APPROXIMATION

 0.3882284792654056D 00 +  0.1448888763117193D 01 I          INITIAL APPROXIMATION
-0.3882284792654056D 00 + -0.1448888763117193D 01 I          ALTERED APPROXIMATION
-0.1060660288248421D 01 +  0.1060660055311209D 01 I          ALTERED APPROXIMATION
 0.1060660288248421D 01 + -0.1060660055311209D 01 I          ALTERED APPROXIMATION
-0.1448888677856240D 01 + -0.3882287974635502D 00 I          ALTERED APPROXIMATION
 0.1448888677856240D 01 +  0.3882287974635502D 00 I          ALTERED APPROXIMATION


COEFFICIENTS OF DEFLATED POLYNOMIAL FOR WHICH NO ZEROS WERE FOUND

  D( 1) =  0.1000000000000000D 01 +  0.0000000000000000D 00 I
  D( 2) =  0.2000000000000000D 01 +  0.0000000000000000 D0 I
  D( 3) = -0.1000000000000000D 01 + -0.0000000000000000 00 I
  D( 4) = -0.2000000000000000D 01 + -0.0000000000000000 00 I
```

Exhibit AA.

```
        NEWTONS METHOD TO FIND ZEROS OF POLYNOMIALS
        POLYNOMIAL NUMBER  1 OF DEGREE  7


THE COEFFICIENTS OF P(X) ARE

   P( 1) =  0.1000000000000000D 01 +  0.0000000000000000D 00 I
   P( 2) = -0.1000000000000000D 01 +  0.1100000000000000D 02 I
   P( 3) = -0.5900000000000001D 02 + -0.2900000000000000D 02 I
   P( 4) =  0.1950000000000000D 03 + -0.1690000000000000D 03 I
   P( 5) =  0.7000000000000001D 02 +  0.7230000000000000D 03 I
   P( 6) = -0.1624000000000000D 04 + -0.6960000000000001D 03 I
   P( 7) =  0.1922000000000000D 04 + -0.1832000000000000D 04 I
   P( 8) =  0.1596000000000000D 04 +  0.1692000000000000D 04 I


NUMBER OF INITIAL APPROXIMATIONS GIVEN.   3
MAXIMUM NUMBER OF ITERATIONS.           200
TEST FOR CONVERGENCE.              0.10D-09
TEST FOR MULTIPLICITIES.           0.10D-01
RADIUS TO START SEARCH.            0.00D 00
RADIUS TO END SEARCH.              0.00D 00


BEFORE THE ATTEMPT TO IMPROVE ACCURACY, THE ZEROS OF P(X) ARE


 ROOTS OF P(X)                                                      MULTIPLICITIES                INITIAL APPROXIMATION


   ROOT( 1) = -0.2999999999999997D 01 + -0.3000000000000002D 01 I        1       -0.3500000000000000D 01 + -0.3500000000000000D 01 I
   ROOT( 2) =  0.2000000000000000D 01 +  0.2000000000000000D 01 I        1        0.2500000000000000D 01 +  0.2500000000000000D 01 I
   ROOT( 3) = -0.9999999999999962D 00 + -0.3999999999999994D 01 I        1       -0.1500000000000000D 01 + -0.4500000000000001D 01 I


AFTER  THE ATTEMPT TO IMPROVE ACCURACY, THE ZEROS OF P(X) ARE


 ROOTS OF P(X)                                                      MULTIPLICITIES                INITIAL APPROXIMATION


   ROOT( 1) = -0.2999999999999996D 01 + -0.3000000000000001D 01 I        1       -0.3500000000000000D 01 + -0.3500000000000000D 01 I
   ROOT( 2) =  0.2000000000000000D 01 +  0.2000000000000000D 01 I        1        0.2500000000000000D 01 +  0.2500000000000000D 01 I
   ROOT( 3) = -0.9999999999999974D 00 + -0.3999999999999996D 01 I        1       -0.1500000000000000D 01 + -0.4500000000000001D 01 I


COEFFICIENTS OF DEFLATED POLYNOMIAL FOR WHICH NO ZEROS WERE FOUND

   D( 1) =  0.1000000000000000D 01 +  0.0000000000000000D 00 I
   D( 2) = -0.2999999999999993D 01 +  0.6000000000000005D 01 I
   D( 3) = -0.2000000000000005D 02 + -0.1899999999999999D 02 I
   D( 4) =  0.4100000000000003D 02 + -0.2200000000000008D 02 I
   D( 5) =  0.2300000000000000D 02 +  0.4100000000000009D 02 I
```

Exhibit BB. Roots Are: -1 - 4i, -2 - 3i, -3 - 3i, -1 - i, 2 + 2i, 4 - i, 2 - i.

```
        NEWTONS METHOD TO FIND ZEROS OF POLYNOMIALS
        POLYNOMIAL NUMBER  1 OF DEGREE  7


THE COEFFICIENTS OF P(X) ARE

  P( 1) =  0.1000000000000000D 01 +  0.0000000000000000D 00 I
  P( 2) = -0.1000000000000000D 01 +  0.1100000000000000D 02 I
  P( 3) = -0.5900000000000001D 02 + -0.2900000000000000D 02 I
  P( 4) =  0.1950000000000000D 03 + -0.1690000000000000D 03 I
  P( 5) =  0.7000000000000001D 02 +  0.7230000000000000D 03 I
  P( 6) = -0.1624000000000000D 04 + -0.6960000000000001D 03 I
  P( 7) =  0.1922000000000000D 04 + -0.1832000000000000D 04 I
  P( 8) =  0.1596000000000000D 04 +  0.1692000000000000D 04 I


NUMBER OF INITIAL APPROXIMATIONS GIVEN.   2
MAXIMUM NUMBER OF ITERATIONS.           200
TEST FOR CONVERGENCE.             0.10D-09
TEST FOR MULTIPLICITIES.          0.10D-01
RADIUS TO START SEARCH.           0.70D 01
RADIUS TO END SEARCH.             0.15D 02


BEFORE THE ATTEMPT TO IMPROVE ACCURACY, THE ZEROS OF P(X) ARE


 ROOTS OF P(X)                                                 MULTIPLICITIES              INITIAL APPROXIMATION

  ROOT( 1) = -0.2999999999999997D 01 + -0.3000000000000002D 01 I      1        -0.3500000000000000D 01 + -0.3500000000000000D 01 I
  ROOT( 2) =  0.2000000000000000D 01 +  0.2000000000000000D 01 I      1         0.2500000000000000D 01 +  0.2500000000000000D 01 I
  ROOT( 3) =  0.4000000000000001D 01 + -0.1000000000000001D 01 I      1         0.6761480761918791D 01 +  0.1811733398213462D 01 I
  ROOT( 4) =  0.1999999999999997D 01 + -0.9999999999999976D 00 I      1         0.5303300664784760D 01 +  0.5303301053013447D 01 I
  ROOT( 5) = -0.9999999999999998D 00 + -0.1000000000000004D 01 I      1         0.2070551889415497D 01 +  0.7727406736625032D 01 I
  ROOT( 6) = -0.9999999999999976D 00 + -0.3999999999999995D 01 I      1         SOLVED BY DIRECT METHOD
  ROOT( 7) = -0.2000000000000004D 01 + -0.3000000000000001D 01 I      1         SOLVED BY DIRECT METHOD


AFTER  THE ATTEMPT TO IMPROVE ACCURACY, THE ZEROS OF P(X) ARE


 ROOTS OF P(X)                                                 MULTIPLICITIES              INITIAL APPROXIMATION

  ROOT( 1) = -0.2999999999999996D 01 + -0.3000000000000001D 01 I      1        -0.3500000000000000D 01 + -0.3500000000000000D 01 I
  ROOT( 2) =  0.2000000000000000D 01 +  0.2000000000000000D 01 I      1         0.2500000000000000D 01 +  0.2500000000000000D 01 I
  ROOT( 3) =  0.4000000000000001D 01 + -0.1000000000000001D 01 I      1         0.6761480761918791D 01 +  0.1811733398213462D 01 I
  ROOT( 4) =  0.2000000000000000D 01 + -0.9999999999999998D 00 I      1         0.5303300664784760D 01 +  0.5303301053013447D 01 I
  ROOT( 5) = -0.9999999999999997D 00 + -0.1000000000000000D 01 I      1         0.2070551889415497D 01 +  0.7727406736625032D 01 I
  ROOT( 6) = -0.9999999999999982D 00 + -0.4000000000000000D 01 I      1         SOLVED BY DIRECT METHOD
  ROOT( 7) = -0.2000000000000003D 01 + -0.3000000000000007D 01 I      1         SOLVED BY DIRECT METHOD
COMPILE TIME=    14.39 SEC,EXECUTION TIME=   103.77 SEC,OBJECT CODE=   18880 BYTES,ARRAY AREA=    3400 BYTES,UNUSED=   47720 BYTES
```

Exhibit CC. Roots Are: $-1 - 4i$, $-2 - 3i$, $-3 - 3i$, $-1 - i$, $2 + 2i$, $4 - i$, $2 - i$.

APPENDIX B

NEWTON'S METHOD

1. Use of the Programs

Two programs using Newton's method are presented here. The first is the single precision program. The second program is in double precision and is designed to perform double precision complex arithmetic. These programs are written for use on any computer using FORTRAN IV language. They have been tested on the IBM S/360 mod. 50 computer which has a 32 bit word. However, it may be necessary to change the system functions as described below. The single precision program may be changed to double precision as described below.

After selecting the desired program, the input data should be prepared as described in section 2.

Each program in designed to solve polynomials of degree 25 or less. Both the coefficient of the highest degree term and the constant coefficient should be non-zero. In order to solve polynomials of degree N, where N > 25, certain array dimensions must be changed. These are listed in Table I for the main program and subprograms in both single precision and double precision.

TABLE I

PROGRAM CHANGES FOR SOLVING POLYNOMIALS OF DEGREE GREATER THAN 25 BY NEWTON'S METHOD

Single Precision	Double Precision
Main Program	
A(N+1)	RA(N+1), VA(N+1)
B(N+1)	RB(N+1), VB(N+1)
C(N+1)	RC(N+1), VC(N+1)
D(N+1)	RD(N+1), VD(N+1)
COEF(N+1)	RCOEF(N+1), VCOEF(N+1)
MULT(N)	MULT(N)
XZERO(N)	RXZERO(N), VXZERO(N)
X(N)	RX(N), VX(N)
XINIT(N)	RXINIT(N), VXINIT(N)
Subroutine HORNER	
A(N+1)	RA(N+1), VA(N+1)
B(N+1)	RB(N+1), VB(N+1)
C(N+1)	RC(N+1), VC(N+1)
Subroutine BETTER	
XZERO(N)	RXZERO(N), VXZERO(N)
X(N)	RX(N), VX(N)
A(N+1)	RA(N+1), VA(N+1)
COEF(N+1)	RCOEF(N+1), VCOEF(N+1)
C(N+1)	RC(N+1), VC(N+1)
B(N+1)	RB(N+1), VB(N+1)
Subroutine GENAPP	
APP(N)	APPR(N), APPI(N)
Subroutine QUAD	
A(N+1)	UA(N+1), VA(N+1)
ROOT(N)	UROOT(N), VROOT(N)
MULTI(N)	MULTI(N)

Certain computers may require that the following system functions in the single precision and double precision programs be changed: A "c" denotes a complex number and an "r" denotes a real number.

TABLE II

SYSTEM FUNCTIONS USED IN NEWTON'S METHOD

Single Precision		Double Precision
CABS(c)	- obtain absolute value -	DABS(r)
COS(r)	- obtain cosine of angle -	DCOS(r)
SIN(r)	- obtain sine of angle -	DSIN(r)
CMPLX(r_1,r_2)	- express two real numbers in complex form	
AIMAG(c)	- obtain imaginary part	
REAL(c)	- obtain real part	
ATAN2(r_1,r_2)	- arctangent of r_1/r_2 -	DATAN2(r_1,r_2)
CSQRT(c)	- square root -	DSQRT(r)

When used on the IBM S/360 with the WATFOR complier for FORTRAN IV, the system functions in Table II-A must be typed in a declaration statement. These also appear in the program listing. For use without the WATFOR compiler or on other computers, these system functions might have to be removed. A "c" denotes a complex number and an "r" denotes a real number.

The single precision program may be converted to double precision for use on machines equipped to perform double precision complex arithmetic provided the following changes or their equivalent are made and the system functions of Table III are used and typed in a declaration statement where necessary. The changes presented below are those required for the IBM S/360. A "c" denotes a complex number and an "r" denotes a real number. The format statements should be changed from E-type to D-type.

In the main program and each subprogram change COMPLEX $c_1, c_2, \ldots$ to COMPLEX*16 $c_1, c_2, \ldots$ and add IMPLICIT REAL*8(A-H,O-Z).

TABLE II-A

SYSTEM FUNCTIONS IN NEWTON'S METHOD TO BE TYPED WHEN THE WATFOR COMPILER IS USED

Single Precision		Double Precision
	Main Program and Subroutines NEWTON, CHECK, and BETTER	
	square root -	DSQRT(r)
	Subroutine GENAPP	
CMPLX(r_1,r_2)	- express in complex form	
	cosine of angle -	DCOS(r)
	sine of angle -	DSIN(r)
	Subroutine ALTER	
CMPLX(r_1,r_2)	- express in complex form	
	cosine of angle -	DCOS(r)
	sine of angle -	DSIN(r)
	arctangent of r_1/r_2 -	DATAN2(r_1,r_2)
	square root -	DSQRT(r)
	Subroutine QUAD	
CSQRT(c)	- square root -	DSQRT(r)
	Subroutine COMSQT	
	absolute value -	DABS(r)
	square root -	DSQRT(r)

TABLE III

SYSTEM FUNCTIONS FOR CONVERTING SINGLE PRECISION NEWTON'S METHOD TO DOUBLE PRECISION

Single Precision	changed to	Double Precision
	Main Program and Subroutines NEWTON, CHECK, BETTER, ALTER, and QUAD	
CABS(c)	- absolute value -	CDABS(c)
	Subroutine GENAPP	
COS(r)	- cosine of angle -	DCOS(r)
SIN(r)	- sine of angle -	DSIN(r)
CMPLX(r_1,r_2)	- express in complex form -	DCMPLX(r_1,r_2)
	Subroutine ALTER	
	Add	COMPLEX*8 SXOLD
CMPLX(r_1,r_2)	- express in complex form -	DCMPLX(r_1,r_2)
AIMAG(c)	- obtain imaginary part -	AIMAG(c)(single precision)
REAL(c)	- obtain real part -	REAL(c)(single precision)
ATAN2(r_1,r_2)	- arctangent of r_1/r_2 -	DATAN2(r_1,r_2)
COS(r)	- cosine of angle -	DCOS(r)
SIN(r)	- sine of angle -	DSIN(r)
	Subroutine QUAD	
CSQRT(c)	- square root -	CDSQRT(c)

2. Input Data for Newton's Method

The input data for Newton's method is grouped into polynomial data sets. Each polynomial data set consists of the data for one and only one polynomial. As many polynomials as the user desires may be solved by placing the polynomial data sets one behind the other. Each polynomial data set consists of three kinds of information placed in the following order:

1. Control information
2. Coefficients of the polynomial
3. Initial approximations. These may be omitted as described in Appendix A, § 1.

An end card follows the entire collection of data sets. It indicates that there is no more data to follow and terminates execution of the program. This information is displayed in Figure 7 and described below. For single precision data, the E-type specification should be used, while for double precision data, the D-type specification should be used. All data should be right justified. The recommendations given in Table IV are those found to give best results on the IBM S/360 mod. 50 computer which has a 32 bit word. The entry in parentheses is the double precision equivalent.

Control Information

The control card is the first card of the polynomial data set and contains the information given in Table IV. See Figure 8.

TABLE IV

CONTROL DATA FOR NEWTON'S METHOD

Variable Name	Card Columns	Description
NOPOLY	c.c. 1-2	Number of the polynomial. Integer. Right justified.
N	c.c. 4-5	Degree of the polynomial. Integer. Right justified.

TABLE IV (Continued)

Variable Name	Card Columns	Description
NIAP	c.c. 7-8	Number of initial approximations to be read. Integer. If no approximations are given, this should be left blank.
MAX	c.c. 19-21	Maximum number of iterations. Integer. Right justified. 200 is recommended.
EPSCNV	c.c. 30-35	Convergence requirement. Real. Right justify. 1.E-05 (1.D-10) is recommended.
EPSQ	c.c. 37-42	Tolerance check for zero (0) in subroutine QUAD. Real. Right justify. 1.E-10 (1.D-20) is recommended.
EPSMUL	c.c. 44-49	Multiplicity requirement. Real. Right justify. 1.E-01 (1.D-02) is recommended.
XSTART	c.c. 64-70	Magnitude at which to begin generating initial approximations. Real. Right justify. This is a special feature of the program and may be omitted.
XEND	c.c. 72-78	Magnitude at which to end the generating of initial approximations. Real. Right justify. This is a special feature of the program and may be omitted.
KCHECK	c.c. 80	This should be left blank.

Coefficients of the Polynomial

The coefficient cards follow the control card. For an N^{th} degree polynomial, N+1 coefficients must be entered one per card. The coefficient of the highest degree term is entered first. For example, if the polynomial $X^5 + 3X^4 + 2X + 5$ were to be solved, the order in which the coefficients would be entered is: 1, 3, 0, 0, 2, 5. Each real or complex coefficient is entered, one per card, as described in Table V and illustrated in Figure 9.

TABLE V

COEFFICIENT DATA FOR NEWTON'S METHOD

Variable Name	Card Columns	Description
A (RA)	c.c. 1-30	Real part of complex coefficient. Real. Right justify. If none, leave blank or enter 0.0E00 (0.0D00).
A (VA)	c.c. 31-60	Imaginary part of complex number. Real. Right justify. If none, leave blank or enter 0.0E00 (0.0D00).

Initial Approximations

The initial approximation cards follow the set of coefficient cards. The number of initial approximations read must be the number specified on the control card and are entered, one per card, as given in Table VI and illustrated in Figure 10.

TABLE VI

INITIAL APPROXIMATION DATA FOR NEWTON'S METHOD

Variable Name	Card Columns	Description
XZERO (RXZERO)	c.c. 1-30	Real part of complex number. Real. Right justify. If none, leave blank or enter 0.0E00 (0.0D00).
XZERO (VXZERO)	c.c. 31-60	Imaginary part of complex number. Real. Right justify. If none, leave blank or enter 0.0E00 (0.0D00).

End Card

The end card is the last card of the input data to the program. It indicates that there is no more data to be read. When this card is read, program execution is terminated. This card is described in Table VII and illustrated in Figure 11.

TABLE VII

DATA TO END EXECUTION OF NEWTON'S METHOD

Variable Name	Card Columns	Description
KCHECK	c.c. 80	Must contain the number 1. Integer.

3. Variables Used in Newton's Method

The definitions at the major variables used in Newton's method are given in Table VIII. The symbols used to indicate type are:

R - real variable

I - integer variable

C - complex variable

L - logical variable

A - alphanumeric variable

When two variables are listed, the one on the left is the real part of the corresponding single precision variable; the one on the right is the imaginary part. The symbols used to indicate disposition are:

E - entered

R - returned

ECR - entered, changed, and returned

C - variable in common

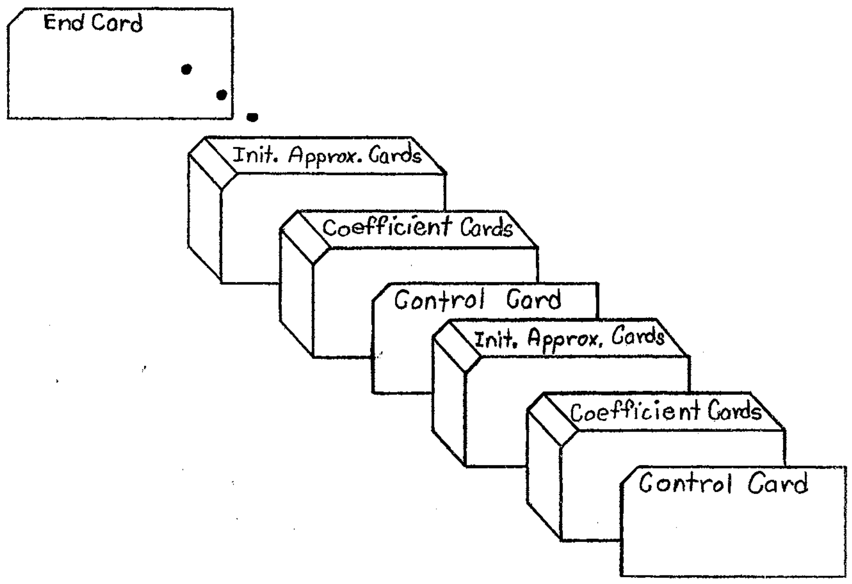
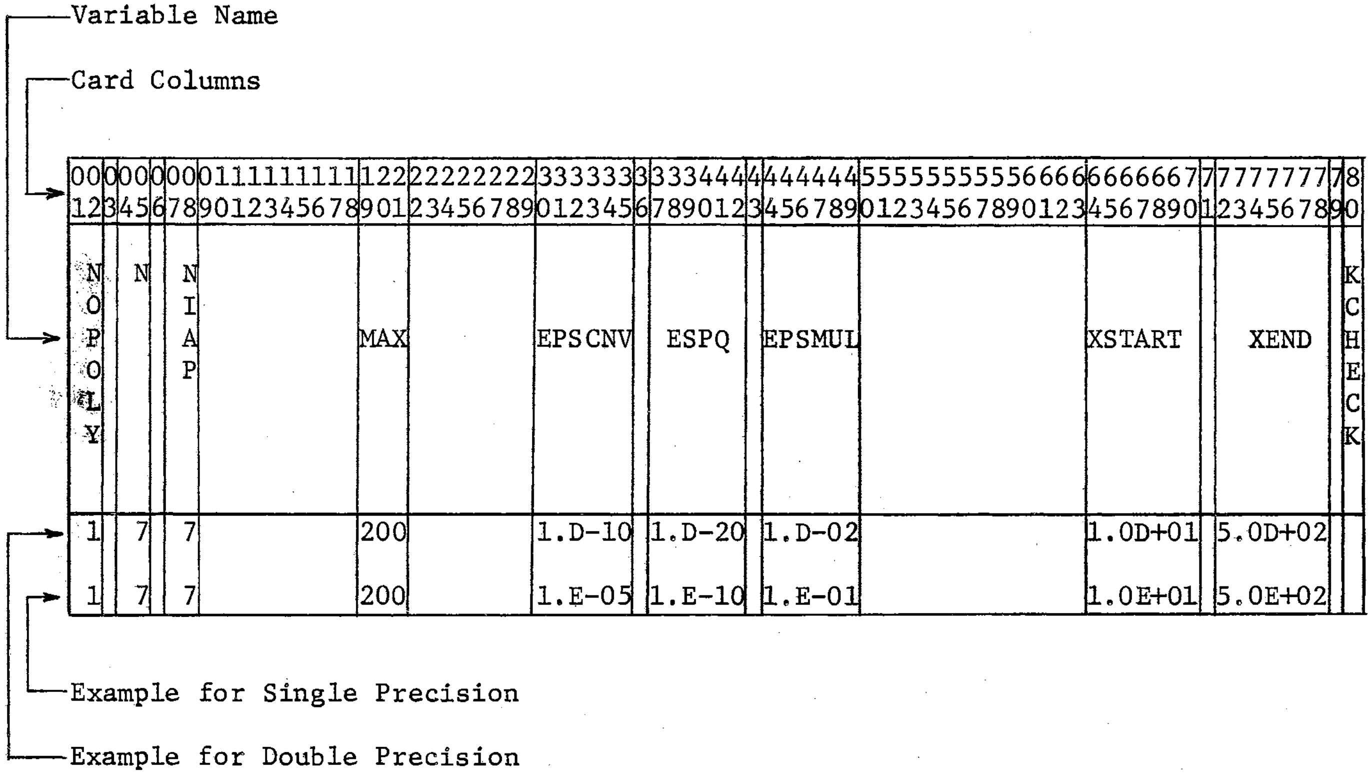


Figure 7. Sequence of Input Data for Newton's Method


Variable Name
Card Columns
Example for Single Precision
Example for Double Precision

Card Columns	00 12	0 3	00 45	0 6	00 78	0111111111 9012345678	122 901	22222222 23456789	333333 012345	3 6	333444 789012	4 3	444444 456789	55555555556666 01234567890123	6666667 4567890	7 1	7777777 2345678	7 9	8 0
Variable Name	NOPOLY		N		NIAP		MAX		EPSCNV		ESPQ		EPSMUL		XSTART		XEND		KCHECK
Example for Double Precision	1		7		7		200		1.D-10		1.D-20		1.D-02		1.0D+01		5.0D+02		
Example for Single Precision	1		7		7		200		1.E-05		1.E-10		1.E-01		1.0E+01		5.0E+02		

Figure 8. Control Card for Newton's Method

000000000111111111122222222223	333333333344444444445555555556	66666666677777777778
123456789012345678901234567890	123456789012345678901234567890	12345678901234567890
A (RA)	A (VA)	
0.621735D+01	-0.132714D-02	
0.621735E+01	-0.132714E-02	

Figure 9. Coefficient Card for Newton's Method

000000000111111111122222222223 123456789012345678901234567890	333333333344444444445555555555 6 123456789012345678901234567890	66666666677777777778 12345678901234567890
XZERO (RXZERO)	XZERO (VXZERO)	
0.15D+01 0.15E+01	-0.25D-00 -0.25E-00	

Figure 10. Initial Approximation Card for Newton's Method

0000000001111111111222222222233333333334444444444555555555566666666667777777777 1234567890123456789012345678901234567890123456789012345678901234567890123456789	8 0
	K C H E C K
	1 1

Figure 11. End Card for Newton's Method

TABLE VIII

VARIABLES USED IN NEWTON'S METHOD

Single Precision		Double Precision		Disposition	
Variable	Type	Variable	Type	of Argument	Description
					Main Program
NOPØLY	I	NOPOLY	I		Number of the polynomial
N	I	N	I		Degree of the polynomial
NIAP	I	NIAP	I		Number of initial approximations to be read
MAX	I	MAX	I		Maximum number of iterations to be performed
EPSCNV	R	EPSCNV	R		Tolerance check for convergence
EPSMUL	R	EPSMUL	R		Tolerance check for multiplicities
EPSQ	R	EPSQ	R		Tolerance check for zero in subroutine QUAD
XSTART	R	XSTART	R		Magnitude from which to begin the search for zeros
XEND	R	XEND	R		Magnitude to end the search for zeros
KCHECK	I	KCHECK	I		Program Control. When KCHECK = 1, program will terminate execution.
NA	I	NA	I		Number of coefficients or original polynomial
A	C	RA,VA	R		Array containing the coefficients of original polynomial P(X)
NDEF	I	NDEF	I		Degree of current deflated polynomial
L	I	L	I		Counter for number of initial approximations used
ITER	I	ITER	I		Counter for number of iterations
NROOT	I	NROOT	I		Counter for number of roots found (counting multiplicities)
IALTER	I	IALTER	I		Counter for number of alterations of each initial approximation
ITIME	I	ITIME	I		Program control
K	I	K	I		Counter for number of distinct roots found
ND	I	ND	I		Program control & number of coefficient of deflated polynomial for which no zeros were found

TABLE VIII (Continued)

Single Precision Variable	Type	Double Precision Variable	Type	Disposition of Argument	Description
XO	C	RXO,VXO	R		Current approximation (X_n) to root
COEF	C	RCOEF,VCOEF	R		Working array containing coefficients of current deflated polynomial
DPX	C	RDPX,VDPX	R		Derivative of P(X) at some value X
PX	C	RPX,VPX	R		Value of P(X) at some point X
XZERO	C	RXZERO, VXZERO	R		Array containing the initial approximations
XNEW	C	RXNEW,VXNEW	R		New approximation (X_{n+1}) obtained from old approximation (X_n) by Newton's Algorithm
KANS	I	KANS	I		KANS = 1 implies convergence, KANS = 0 implies no convergence
MULT	I	MULT	I		Array containing the number of multiplicities of each root
X	C	RX,VX	R		Array containing the zeros of P(X)
XINIT	C	RXINIT, VXINIT	R		Array containing the initial or altered approximations which produced convergence to each root
NUM	I	NUM	I		Number of coefficients of current deflated polynomial
B	C	RB,VB	R		Array containing the coefficients of newly deflated polynomial
IROOT	I	IROOT	I		Number of distinct roots found by Newton's method, i.e. not solved for directly by subroutine QUAD
D	C	RD,VD	R		Array containing the coefficients of deflated polynomial for which no zeros were found
IO1	I	IO1	I		Unit number of input device
IO2	I	IO2	I		Unit number of output device
C	C	RC,VC	R		Array containing sequence of values leading to the derivative
EPSCHK	R	EPSCHK	R		Current tolerance for checking convergence or multiplicity

TABLE VIII (Continued)

Single Precision Variable	Type	Double Precision Variable	Type	Disposition of Argument	Description
					Subroutine HORNER
A	C	RA,VA	R	E	Array of coefficients of polynomial
B	C	RB,VB	R	R	Array of coefficients of deflated polynomial
NDEF	I	NDEF	I	E	Degree of polynomial
NUM	I	NUM	I		Number of coefficients of polynomial
XO	C	RXO,VXO	R	E	Point (X_n) at which to evaluate the polynomial and its derivative. Also current approximation (X_{n+1}) used to deflate the polynomial
PX	C	RPX,VPX	R	R	Value of polynomial at X_n
DPX	C	RDPX,VDPX	R	R	Value of the derivative of polynomial at X_n
C	C	RC,VC	R	R	Array of containing sequence of values leading to the derivative
					Subroutine NEWTON
PX	C	RPX,VPX	R	E	Value of polynomial at X_n
DPX	C	RDPX,VDPX	R	E	Derivative of polynomial at X_n
XO	C	RXO,VXO	R	E	Current approximation (X_n) to root
XNEW	C	RXNEW,VXNEW	R	R	New approximation (X_{n+1}) to root
					Subroutine CHECK
EPSLON	R	EPS	R	C	Tolerance for convergence or multiplicity check
PX	C	RPX,VPX	R	E	Value of P(X) at X_n
DPX	C	RDPX,VDPX	R	E	Derivative of P(X) at X_n
XO	C	RXO,VXO	R	E	Current approximations (X_{n+1}) to root
IO2	I	IO2	I	C	Unit number of output device
KANS	I	KANS	I	R	KANS = 1 implies convergence, KANS = 0 implies no convergence

TABLE VIII (Continued)

Single Precision Variable	Single Precision Type	Double Precision Variable	Double Precision Type	Disposition of Argument	Description
					Subroutine BETTER
IO2	I	IO2	I	C	Unit number of output device
XZERO	C	RXZERO, VXZERO	R	E	Array of approximations
X	C	RX,VX	R	ECR	Array of roots
A	C	RA,VA	R	E	Coefficients of original (undeflated) polynomial, P(X)
COEF	C	RCOEF,VCOEF	R	E	Working array for coefficients of polynomial
NA	I	NA	I	E	Number of coefficients of original polynomial
XO	C	RXO,VXO	R		Current approximation (X_n) to root
DPX	C	RDPX,VDPX	R		Derivative of P(X) at X_n
PX	C	RPX,VPX	R		Value of P(X) at X_n
KANS	I	KANS	I		KANS = 1 implies convergence; KANS = 0 implies no convergence
ITER	I	ITER	I		Counter for number of iterations
XNEW	C	RXNEW,VXNEW	R		New approximation (X_{n+1}) to root
NN	I	NN	I		Degree of polynomial
C	C	RC,VC	R	E	Array containing the sequence of values leading to the derivative
K	I	K	I	E	Number of distinct roots of P(X) found
N	I	N	I	E	Degree of polynomial P(X)
B	C	RB,VB	R	E	Array of coefficients of deflated polynomial
MAX	I	MAX	I	C	Maximum number of iterations permitted
EPSCHK	R	EPS	R	C	Tolerance for checking convergence
					Subroutine GENAPP
APP	C	APPR,APPI	R	R	Array containing initial approximations
NAPP	I	NAPP	I	E	Number of initial approximations to be generated

TABLE VIII (Continued)

Single Precision Variable	Single Precision Type	Double Precision Variable	Double Precision Type	Disposition of Argument	Description
XSTART	R	XSTART	R	ECR	Magnitude at which to begin generating approximations; also magnitude of the approximation being generated
BETA	R	BETA	R		Argument of the complex approximation being generated
U	R	APPR(I)	R		Real part of complex approximation
V	R	APPI(I)	R		Imaginary part. of complex approximation
					Subroutine ALTER
XOLD	C	XOLDR,XOLDI	R	ECR	Old approximation to be altered to new approximation
NALTER	I	NALTER	I	ECR	Number of alterations performed on an initial approximation
ITIME	I	ITIME	I	E	Program control
MAX	I	MAX	I	C	Maximum number of iterations permitted
Y	R	XOLDI	R		Imaginary part of original initial approximation (unaltered)
X	R	XOLDR	R		Real part of original unaltered initial approximation
R	R	R	R		Magnitude of original unaltered initial approximation
BETA	R	BETA	R		Argument of new approximation
XOLDR	R	XOLDR	R		Real part of new approximation
XOLDI	R	XOLDI	R		Imaginary part of new approximation
IO2	I	IO2	I	C	Unit number of output device
					Subroutine QUAD
A	C	UA,VA	R	E	Coefficients of polynomial to be solved
NA	I	NA	I	E	Degree of polynomial
ROOT	C	UROOT,VROOT	R	ECR	Array of roots of P(X) (original polynomial)
NROOT	I	NROOT	I	ECR	Number of distinct roots of P(X) (the original polynomial)

TABLE VIII (Continued)

Single Precision Variable	Single Precision Type	Double Precision Variable	Double Precision Type	Disposition of Argument	Description
MULTI	I	MULTI	I	ECR	Array containing multiplicities of each root
EPST	R	EPST	R	E	Tolerance check for the number zero
DISC	C	UDISC,VDISC	R		Value of the discriminate ($b^2 - 4ac$) of Quadratic
				Subroutine COMSQT	
		UX,VX	R	E	Complex number for which the square root is desired
		UY,VY	R	R	Square root of the complex number

4. Description of Program Output

The output from Newton's method programs consist of the following information.

The number and degree of the polynomial are printed in the heading (exhibit B, Chapter VII).

The coefficients are printed under the heading "THE COEFFICIENTS OF P(X) ARE." The coefficient of the highest degree term is listed first (exhibit B, Chapter VII).

As an aid to ensure the control information is correct, the number of initial approximations given, maximum number of iterations, test for convergence, test for multiplicities, radius to start search, and radius to end search are printed as read from the control card (exhibit B, Chapter VII).

The zeros found before and after the attempt to improve accuracy are printed. See Apepndix A, § 4 for further explanation (exhibit B, Chapter VII).

If not all zeros of the polynomial are found, the coefficients of the remaining unsolved polynomial will be printed, with coefficient of highest degree term first, under the heading "COEFFICIENTS OF DEFLATED POLYNOMIAL FOR WHICH NO ZEROS WERE FOUND." See Appendix A, § 6. This is illustrated in exhibit BB of Appendix A.

The multiplicity of each zero is given under the title "MULTIPLICITIES" (exhibit B, Chapter VII).

The initial approximation producing convergence to a root is printed to the right of the corresponding root and headed by "INITIAL APPROXIMATION." The initial approximations may be those supplied by the user, or generated by the program, or a combination of both (exhibit CC, Appendix A). See Appendix A, § 1 and § 2 for discussion of approximations. The message "SOLVED BY DIRECT METHOD" indicates that the corresponding root or roots was obtained by Subroutine QUAD. See Appendix A, § 5.

If an approximation does not produce convergence within the maximum number of iterations, it is printed under the heading "NO CONVERGENCE FOR THE FOLLOWING APPROXIMATIONS AFTER XXX ITERATIONS." XXX is replaced by the maximum number of iterations. The type of the approximation, that is, initial approximation or altered approximations is given (exhibit AA, Appendix A). See Appendix A, § 1 and § 2 for discussion of approximations.

5. Informative and Error Messages

The output may contain informative or error messages. These are intended as an aid to the user and are described as follows:

"IN THE ATTEMPT TO IMPROVE ACCURACY, ROOT(X) = YYY DID NOT CONVERGE. THE PRESENT APPROXIMATION AFTER ZZZ ITERATIONS IS PRINTED BELOW." X is the number of the zero, YYY is the value of the zero before the attempt to improve accuracy, ZZZ is the maximum number of iterations. This message indicates that a zero found before attempting to improve accuracy did not converge sufficiently when being used as an initial approximation on the full (undeflated) polynomial. The current approximation is printed in the list of improved zeros. In many cases, this failure to converge is a result of an ill-conditioned polynomial and this current approximation of the root may be better than its approximation before the attempt to improve accuracy. In most cases, the polynomial from which this root was first extracted had fewer multiple roots, due to deflations, than the original polynomial.

"THE VALUE OF THE DERIVATIVE AT XO = XXX IS ZERO."

This message is printed as a result of the value of the derivative of the original polynomial at an approximation, XXX, being zero (0). It occurred in the attempt to improve the accuracy of a zero. The previous message is then printed.

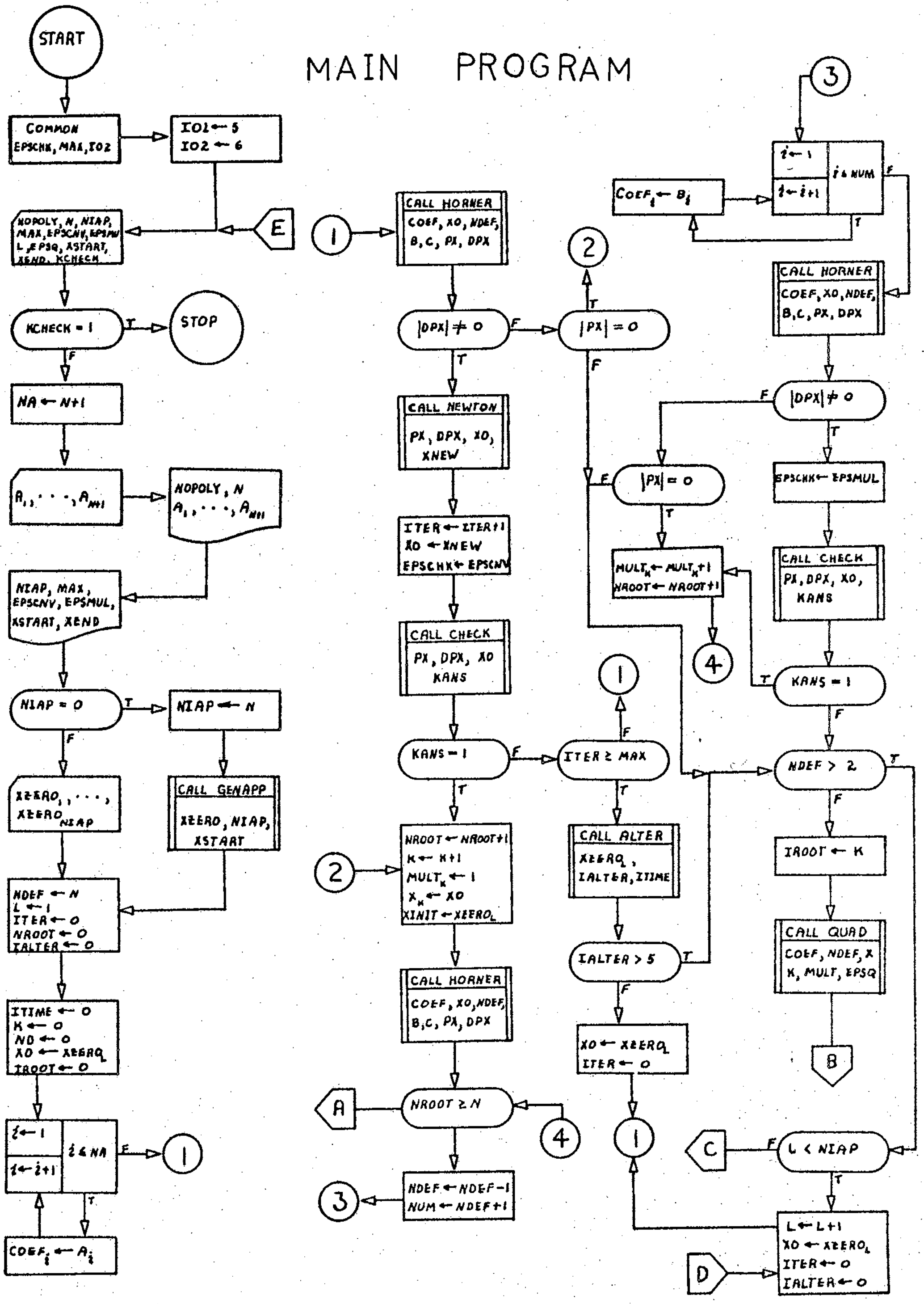


Figure 12. Flow Charts for Newton's Method

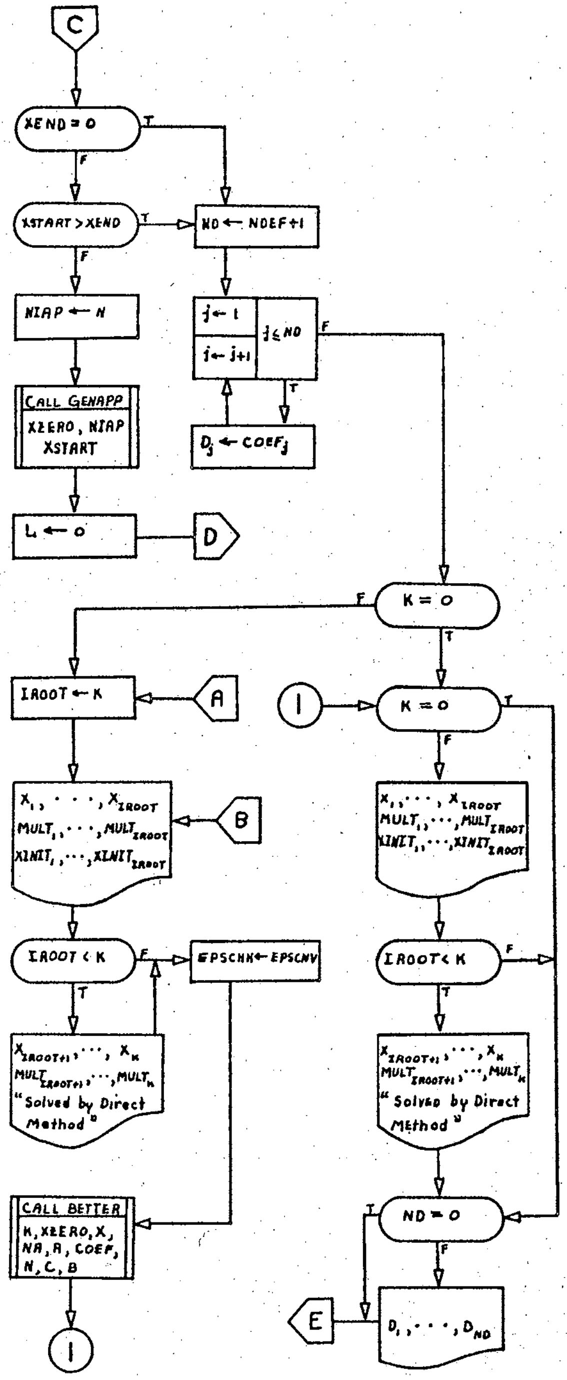
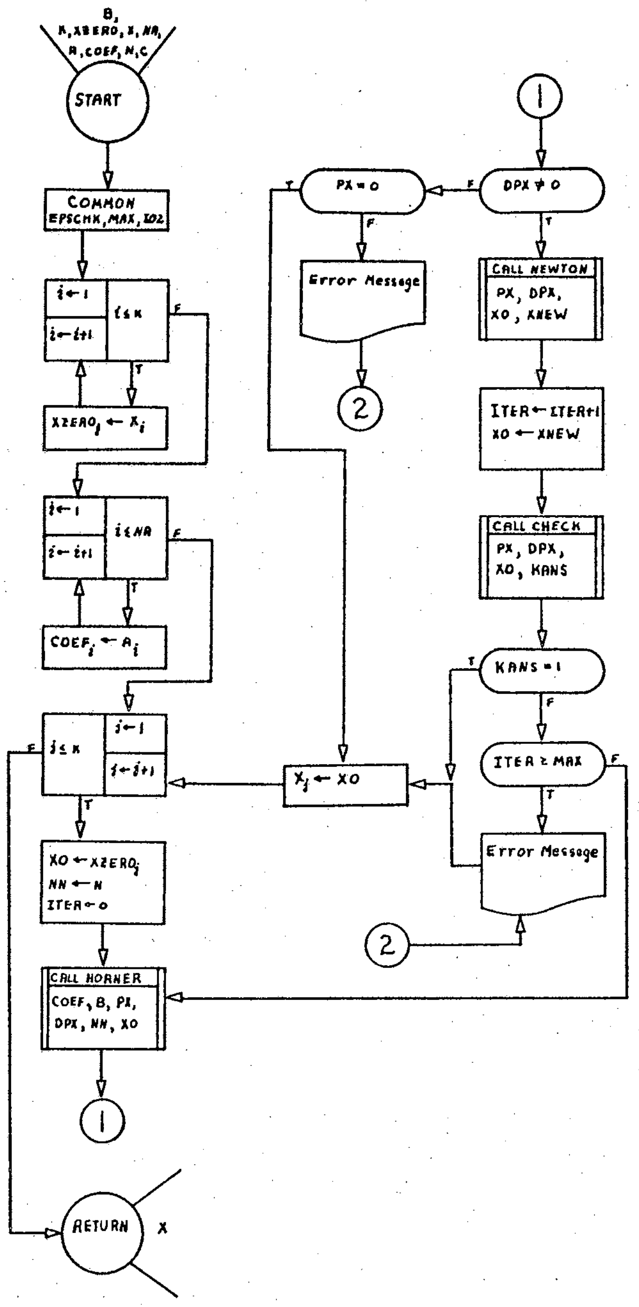


Figure 12. (Continued)

BETTER



CHECK

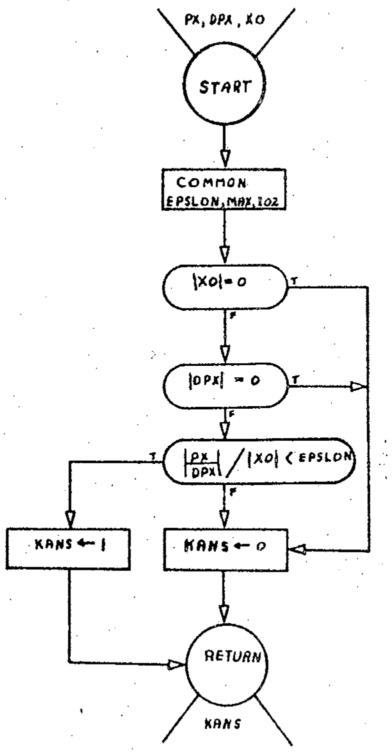


Figure 12. (Continued)

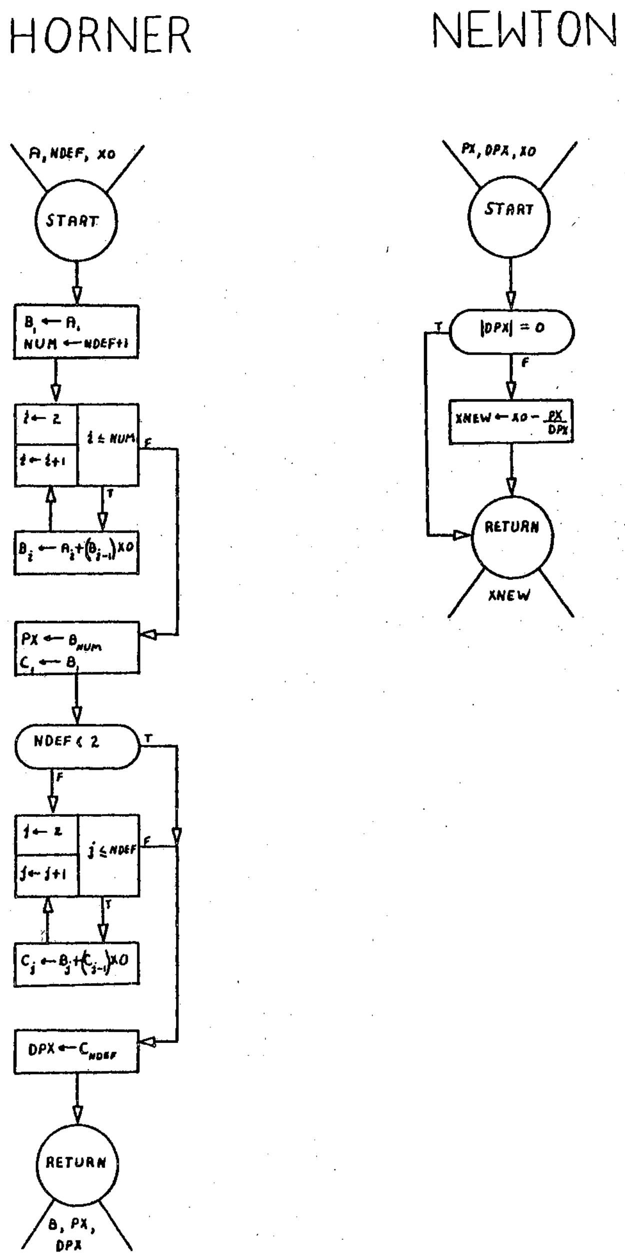
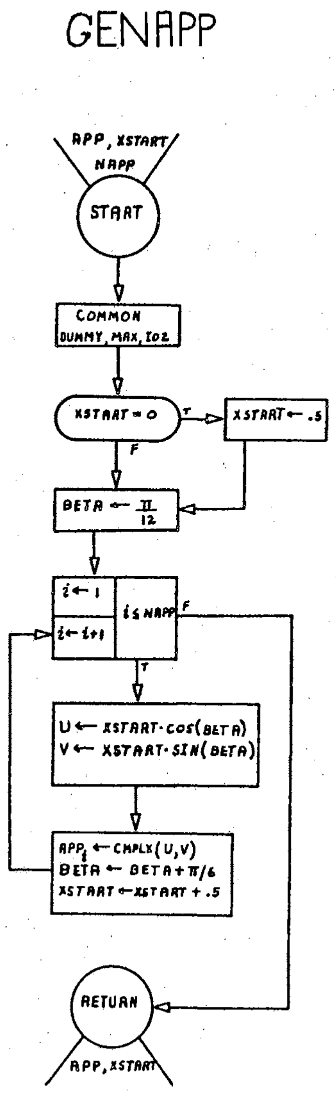


Figure 12. (Continued)



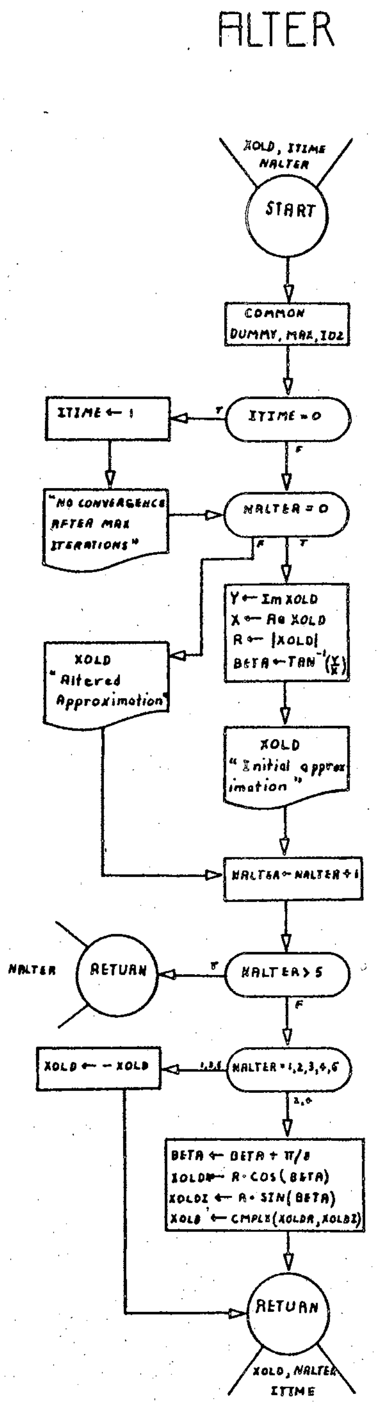
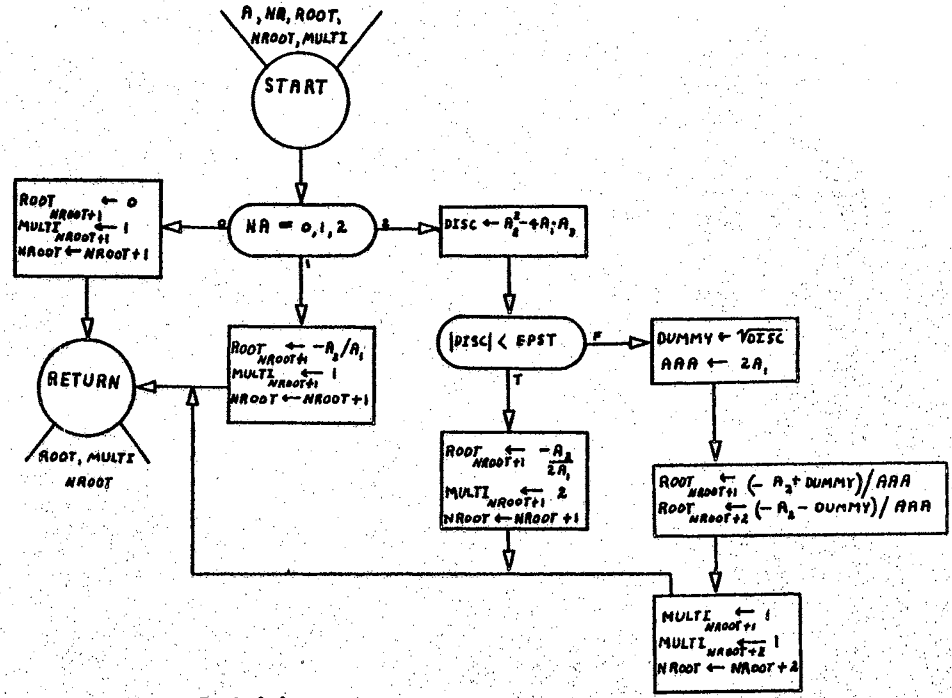


Figure 12. (Continued)

QUAD


A, NA, ROOT, NROOT, MULTI
START
NA = 0, 1, 2
0
1
2
$ROOT_{NROOT+1} \leftarrow 0$
$MULTI_{NROOT+1} \leftarrow 1$
$NROOT \leftarrow NROOT+1$
$DISC \leftarrow A_2^2 - 4A_1 A_3$
$ROOT_{NROOT+1} \leftarrow -A_2/A_1$
$MULTI_{NROOT+1} \leftarrow 1$
$NROOT \leftarrow NROOT+1$
|DISC| < EPST
F
T
$DUMMY \leftarrow \sqrt{DISC}$
$AAA \leftarrow 2A_1$
RETURN
ROOT, MULTI, NROOT
$ROOT_{NROOT+1} \leftarrow -\frac{A_2}{2A_1}$
$MULTI_{NROOT+1} \leftarrow 2$
$NROOT \leftarrow NROOT+1$
$ROOT_{NROOT+1} \leftarrow (-A_2 + DUMMY)/AAA$
$ROOT_{NROOT+2} \leftarrow (-A_2 - DUMMY)/AAA$
$MULTI_{NROOT+1} \leftarrow 1$
$MULTI_{NROOT+2} \leftarrow 1$
$NROOT \leftarrow NROOT+2$

COMSQT

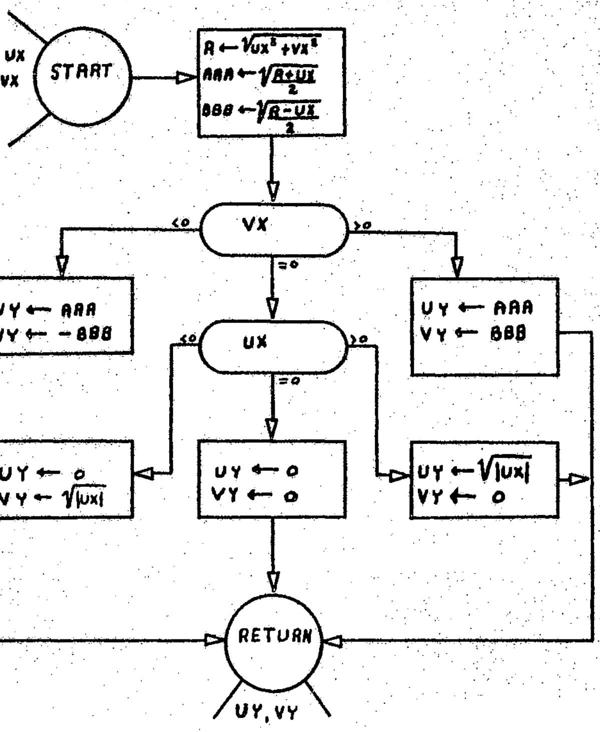


Figure 12. (Continued)

TABLE VIII-A

SINGLE PRECISION PROGRAM FOR NEWTON'S METHOD

```
$JOB 10414
C    ***********************************************************************
C    *                                                                     *
C    * SINGLE PRECISION PROGRAM FOR NEWTON'S METHOD                        *
C    *                                                                     *
C    *                                                                     *
C    * NEWTONS METHOD EXTRACTS THE ZEROS AND THEIR MULTIPLICITIES OF A     *
C    * POLYNOMIAL OF MAXIMUM DEGREE 25 BY COMPUTING A SEQUENCE OF APPROX-  *
C    * IMATIONS CONVERGING TO A ZERO OF THE POLYNOMIAL USING THE ITERATION *
C    * FORMULA                                                             *
C    *               X(N+1) = X(N)-P(X(N))/P'(X(N)).                       *
C    *                                                                     *
C    ***********************************************************************
        COMPLEX A,XZERO,B,COEF,X,XINIT,C,D,PX,DPX,XNEW,XO
        DIMENSION A(26),B(26),C(26),D(26),COEF(26),MULT(25),XZERO(25),X(25
       1),XINIT(25)
        COMMON EPSCHK,MAX,IO2
        IO1=5                                                              112
        IO2=6                                                              116
      1 READ(IO1,1000) NOPOLY,N,NIAP,MAX,EPSCNV,EPSQ,EPSMUL,XSTART,XEND,KC
       1HECK
        IF(KCHECK.EQ.1) STOP
        NA=N+1                                                             130
        READ(IO1,1010) (A(I),I=1,NA)                                       132
        WRITE(IO2,1030) NOPOLY,N
        WRITE(IO2,1040) (I,A(I),I=1,NA)
        WRITE(IO2,2060)
        WRITE(IO2,2000) NIAP
        WRITE(IO2,2010) MAX
        WRITE(IO2,2020) EPSCNV
        WRITE(IO2,2030) EPSMUL
        WRITE(IO2,2040) XSTART
        WRITE(IO2,2050) XEND
        IF(NIAP.NE.0) GO TO 3
        NIAP=N
        CALL GENAPP(XZERO,NIAP,XSTART)
        GO TO 4
      3 READ(IO1,1020) (XZERO(I),I=1,NIAP)
      4 NDEF=N
        L=1                                                                152
        ITER=0                                                             154
        NROOT=0                                                            160
        IROOT=0
        IALTER=0                                                           164
        ITIME=0
        ND=0
        K=0                                                                168
        XO=XZERO(L)                                                        180
        DO 5 I=1,NA                                                        188
      5 COEF(I)=A(I)                                                       190
     10 CALL HORNER(COEF,B,PX,DPX,NDEF,XO,C)
        ABPX=CABS(PX)
        ABDPX=CABS(DPX)
        IF(ABDPX.NE.0.0) GO TO 20
        IF(ABPX.EQ.0.0) GO TO 70
        GO TO 110
     20 CALL NEWTON(PX,DPX,XO,XNEW)                                        198
        ITER=ITER+1                                                        200
        XO=XNEW                                                            204
```

TABLE VIII-A (Continued)

```
      EPSCHK=EPSCNV
      CALL CHECK(PX,DPX,XO,KANS)
      IF(KANS.EQ.1) GO TO 70                                       194
      IF(ITER.GE.MAX) GO TO 40
      GO TO 10                                                     208
   40 CALL ALTER(XZERO(L),IALTER,ITIME)
      IF(IALTER.GT.5) GO TO 110
      XO=XZERO(L)
      ITER=0                                                       244
      GO TO 10                                                     248
   60 ND=NDEF+1
      DO 65 J=1,ND
   65 D(J)=COEF(J)
      GO TO 140
   70 NROOT=NROOT+1                                                268
      K=K+1                                                        272
      MULT(K)=1                                                    276
      X(K)=XO                                                      280
      XINIT(K)=XZERO(L)                                            288
      CALL HORNER(COEF,B,PX,DPX,NDEF,XO,C)
   80 IF(NROOT.GE.N) GO TO 147
      NDEF=NDEF-1
      NUM=NDEF+1
      DO 105 I=1,NUM                                               294
  105 COEF(I)=B(I)                                                 296
      CALL HORNER(COEF,B,PX,DPX,NDEF,XO,C)
      ABPX=CABS(PX)
      ABDPX=CABS(DPX)
      IF(ABDPX.NE.0.0) GO TO 107
      IF(ABPX.EQ.0.0) GO TO 130
      GO TO 110
  107 CONTINUE
      EPSCHK=EPSMUL
      CALL CHECK(PX,DPX,XO,KANS)
      IF(KANS.EQ.1) GO TO 130                                      300
  110 IF(NDEF.GT.2) GO TO 113
      IROOT=K
      CALL QUAD(COEF,NDEF,X,K,MULT,EPSQ)
      GO TO 150
  113 IF(L.LT.NIAP) GO TO 115
      IF(XEND.EQ.0.0) GO TO 60
      IF(XSTART.GT.XEND) GO TO 60
      NIAP=N
      CALL GENAPP(XZERO,NIAP,XSTART)
      L=0
  115 L=L+1
      XO=XZERO(L)                                                  312
      ITER=0                                                       316
      IALTER=0                                                     320
      GO TO 10                                                     324
  130 MULT(K)=MULT(K)+1                                            328
      NROOT=NROOT+1                                                332
      GO TO 80                                                     336
  140 IF(K.EQ.0) GO TO 160                                         338
  147 IROOT=K
  150 WRITE(IO2,1025)
      WRITE(IO2,1050)                                              380
      WRITE(IO2,1060) (I,X(I),MULT(I),XINIT(I),I=1,IROOT)
      KKK=IROOT+1
      IF(IROOT.LT.K) WRITE(IO2,1062) (I,X(I),MULT(I),I=KKK,K)
```

TABLE VIII-A (Continued)

```
      EPSCHK=EPSCNV
      CALL BETTER(K,XZERO,X,NA,A,COEF,N,C,B)
  160 IF(K.EQ.0) GO TO 170                                                  378
      WRITE(IO2,1065)
      WRITE(IO2,1050)
      WRITE(IO2,1060) (I,X(I),MULT(I),XINIT(I),I=1,IROOT)
      KKK=IROOT+1
      IF(IROOT.LT.K) WRITE(IO2,1062) (I,X(I),MULT(I),I=KKK,K)
  170 IF(ND.EQ.0) GO TO 1
      WRITE(IO2,1070)                                                       396
      WRITE(IO2,1075) (J,D(J),J=1,ND)
      GO TO 1                                                               444
 1000 FORMAT(3(I2,1X),9X,I3,8X,3(E6.0,1X),13X,2(E7.0,1X),I1)
 1010 FORMAT(2E30.0)
 1030 FORMAT(1H1,8X,43HNEWTONS METHOD TO FIND ZEROS OF POLYNOMIALS/9X,18
     1HPOLYNOMIAL NUMBER ,I2,11H OF DEGREE ,I2,////1X,28HTHE COEFFICIENT
     2S OF P(X) ARE/)
 1040 FORMAT(3X,2HP(,I2,4H) = ,E14.7,3H + ,E14.7,2H I)
 2000 FORMAT(1X,41HNUMBER OF INITIAL APPROXIMATIONS GIVEN.  ,I2)
 2010 FORMAT(1X,29HMAXIMUM NUMBER OF ITERATIONS.,11X,I3)
 2020 FORMAT(1X,21HTEST FOR CONVERGENCE.,13X,E9.2)
 2030 FORMAT(1X,24HTEST FOR MULTIPLICITIES.,10X,E9.2)
 2040 FORMAT(1X,23HRADIUS TO START SEARCH.,11X,E9.2)
 2050 FORMAT(1X,21HRADIUS TO END SEARCH.,13X,E9.2)
 2060 FORMAT(//1X)
 1020 FORMAT(2E30.0)
 1025 FORMAT(///1X,61HBEFORE THE ATTEMPT TO IMPROVE ACCURACY, THE ZEROS
     1OF P(X) ARE)
 1050 FORMAT(///1X,13HZEROS OF P(X),38X,14HMULTIPLICITIES,11X,21HINITIAL
     1 APPROXIMATION/)
 1060 FORMAT(3X,5HROOT(,I2,4H) = ,E14.7,3H + ,E14.7,2H I,9X,I2,12X,E14.7
     1,3H + ,E14.7,2H I)
 1062 FORMAT(3X,5HROOT(,I2,4H) = ,E14.7,3H + ,E14.7,2H I,9X,I2,13X,23HSO
     1LVED BY DIRECT METHOD)
 1065 FORMAT(///1X,61HAFTER  THE ATTEMPT TO IMPROVE ACCURACY, THE ZEROS
     1OF P(X) ARE)
 1070 FORMAT(///1X,65HCOEFFICIENTS OF DEFLATED POLYNOMIAL FOR WHICH NO Z
     1EROS WERE FOUND/)
 1075 FORMAT(3X,2HD(,I2,4H) = ,E14.7,3H + ,E14.7,2H I)
      END                                                                   450
```

TABLE VIII-A (Continued)

```
      SUBROUTINE HORNER(A,B,PX,DPX,NDEF,XO,C)
C     ************************************************************************
C     *                                                                      *
C     * HORNER'S METHOD COMPUTES THE VALUE OF A POLYNOMIAL P(X) AT A POINT D AND *
C     * ITS DERIVATIVE AT D.  SYNTHETIC DIVISION IS USED TO DEFLATE THE      *
C     * POLYNOMIAL BY DIVIDING OUT THE FACTOR (X-D).                         *
C     *                                                                      *
C     ************************************************************************
      COMPLEX A,B,PX,DPX,XO,C                                             504
      DIMENSION A(26),B(26),C(26)
      B(1)=A(1)                                                           516
      NUM=NDEF+1                                                          520
      DO 10 I=2,NUM                                                       524
   10 B(I)=A(I)+B(I-1)*XO                                                 528
      PX=B(NUM)                                                           532
      C(1)=B(1)                                                           540
      IF(NDEF.LT.2) GO TO 25
      DO 20 J=2,NDEF                                                      544
   20 C(J)=B(J)+C(J-1)*XO                                                 548
   25 DPX=C(NDEF)
      RETURN                                                              572
      END                                                                 580
```

```
      SUBROUTINE NEWTON(PX,DPX,XO,XNEW)                                   600
C     ************************************************************************
C     *                                                                      *
C     * THIS SUBROUTINE CALCULATES A NEW APPROXIMATION FROM THE OLD APPROX-  *
C     * IMATION BY USING THE ITERATION FORMULA                               *
C     *               X(N+1) = X(N)-P(X(N))/P'(X(N)).                        *
C     *                                                                      *
C     ************************************************************************
      COMPLEX PX,DPX,XO,XNEW                                              604
      DDD=CABS(DPX)
      IF(DDD.EQ.0.0) RETURN
      XNEW=XO-(PX/DPX)                                                    612
      RETURN                                                              616
      END                                                                 620
```

TABLE VIII-A (Continued)

```
      SUBROUTINE CHECK(PX,DPX,XO,KANS)
C     ************************************************************************
C     *                                                                      *
C     * THIS SUBROUTINE CHECKS FOR CONVERGENCE OF THE SEQUENCE OF APPROX-    *
C     * IMATIONS BY TESTING THE EXPRESSION                                   *
C     * ABSOLUTE VALUE OF (P(X(N))/P'(X(N)))/ABSOLUTE VALUE OF X(N+1).       *
C     * WHEN IT IS AS SMALL AS DESIRED, CONVERGENCE IS OBTAINED.             *
C     *                                                                      *
C     ************************************************************************
      COMPLEX PX,DPX,XO                                                    748
      COMMON EPSLON,MAX,IO2
      IF(CABS(XO).EQ.0.) GO TO 25
      DDD=CABS(DPX)
      IF(DDD.EQ.0.0) GO TO 25
      IF(CABS(PX/DPX)/CABS(XO).LT.EPSLON) GO TO 10
      KANS=0                                                               760
      RETURN                                                               764
   10 KANS=1                                                               768
      RETURN                                                               772
   25 KANS=0
      RETURN
      END                                                                  780
```

TABLE VIII-A (Continued)

```
        SUBROUTINE BETTER(K,XZERO,X,NA,A,COEF,N,C,B)
C     ************************************************************************
C     *                                                                      *
C     * SUBROUTINE BETTER ATTEMPTS TO IMPROVE THE ACCURACY OF THE ZEROS FOUND *
C     * BY USING THEM AS INITIAL APPROXIMATIONS WITH NEWTON'S METHOD APPLIED TO *
C     * THE FULL, UNDEFLATED POLYNOMIAL.                                     *
C     *                                                                      *
C     ************************************************************************
        COMPLEX XZERO,X,A,COEF,C,B,XO,PX,DPX,XNEW                         804
        DIMENSION XZERO(25),X(25),A(26),COEF(26),C(26),B(26)
        COMMON EPSCHK,MAX,IO2
        DO 10 I=1,K                                                       812
     10 XZERO(I)=X(I)                                                     815
        DO 20 I=1,NA                                                      820
     20 COEF(I)=A(I)                                                      824
        DO 50 J=1,K                                                       828
        XO=XZERO(J)                                                       832
        NN=N                                                              834
        ITER=0                                                            836
     30 CALL HORNER(COEF,B,PX,DPX,NN,XO,C)
        ABPX=CABS(PX)
        ABDPX=CABS(DPX)
        IF(ABDPX.NE.0.0) GO TO 33
        IF(ABPX.EQ.0.0) GO TO 40
        GO TO 34
     33 CALL NEWTON(PX,DPX,XO,XNEW)
        ITER=ITER+1                                                       856
        XO=XNEW                                                           860
        CALL CHECK(PX,DPX,XO,KANS)
        IF(KANS.EQ.1) GO TO 40                                            844
        IF(ITER.GE.MAX) GO TO 35
        GO TO 30                                                          864
     34 WRITE(IO2,1112) XO
     35 WRITE(IO2,100) J,XZERO(J)
        WRITE(IO2,200) MAX
     40 X(J)=XO                                                           868
     50 CONTINUE                                                          872
        RETURN                                                            876
   1112 FORMAT(1H0,36HTHE VALUE OF THE DERIVATIVE AT XO = ,E14.7,3H + ,E14
       1.7,2H I,10H  IS ZERO.)
    100 FORMAT(42H0IN THE ATTEMPT TO IMPROVE ACCURACY, ROOT(,I2,4H) = ,E14
       1.7,3H + ,E14.7,2H I,18H DID NOT CONVERGE.)
    200 FORMAT(33H THE PRESENT APPROXIMATION AFTER ,I3,29H ITERATIONS IS P
       1RINTED BELOW.)
        END                                                               880
```

TABLE VIII-A (Continued)

```
      SUBROUTINE GENAPP(APP,NAPP,XSTART)
C     ****************************************************************************
C     *                                                                          *
C     * SUBROUTINE GENAPP GENERATES N INITIAL APPROXIMATIONS, WHERE N IS THE     *
C     * DEGREE OF THE ORIGINAL POLYNOMIAL.                                       *
C     *                                                                          *
C     ****************************************************************************
      COMPLEX APP
      COMPLEX CMPLX
      DIMENSION APP(25)
      COMMON DUMMY,MAX,IO2
      IF(XSTART.EQ.0.0) XSTART=0.5
      BETA=0.2617994
      DO 10 I=1,NAPP
      U=XSTART*COS(BETA)
      V=XSTART*SIN(BETA)
      APP(I)=CMPLX(U,V)
      BETA=BETA+0.5235988
   10 XSTART=XSTART+0.5
      RETURN
      END




      SUBROUTINE ALTER(XOLD,NALTER,ITIME)
C     ****************************************************************************
C     *                                                                          *
C     * SUBROUTINE ALTER ALTERS THE INITIAL APPROXIMATIONS WHICH PRODUCE NO      *
C     * CONVERGENCE TO A ZERO. THIS IS DONE A MAXIMUM OF 5 TIMES FOR EACH ROOT.  *
C     *                                                                          *
C     ****************************************************************************
      COMPLEX XOLD
      COMPLEX CMPLX
      COMMON DUMMY,MAX,IO2
      IF(ITIME.NE.0) GO TO 5
      ITIME=1
      WRITE(IO2,1010) MAX
    5 IF(NALTER.EQ.0) GO TO 10
      WRITE(IO2,1000) XOLD
      GO TO 20
   10 Y=AIMAG(XOLD)
      X=REAL(XOLD)
      R=CABS(XOLD)
      BETA=ATAN2(Y,X)
      WRITE(IO2,1020) XOLD
   20 NALTER=NALTER+1
      IF(NALTER.GT.5) RETURN
      GO TO (30,40,30,40,30),NALTER
   30 XOLD=-XOLD
      GO TO 50
   40 BETA=BETA+1.0471976
      XOLDR=R*COS(BETA)
      XOLDI=R*SIN(BETA)
      XOLD=CMPLX(XOLDR,XOLDI)
   50 RETURN
 1000 FORMAT(1X,E14.7,3H + ,E14.7,2H I,10X,21HALTERED APPROXIMATION)
 1010 FORMAT(///1X,54HNO CONVERGENCE FOR THE FOLLOWING APPROXIMATIONS AF
     1TER ,I3,12H ITERATIONS.//)
 1020 FORMAT(/1X,E14.7,3H + ,E14.7,2H I,10X,21HINITIAL APPROXIMATION)
      END
```

TABLE VIII-A (Continued)

```
      SUBROUTINE QUAD(A,NA,ROOT,NROOT,MULTI,EPST)
C    *****************************************************************************
C    *                                                                           *
C    * SUBROUTINE QUAD SOLVES DIRECTLY FOR THE ZEROS AND THEIR MULTIPLICITIES   *
C    * OF EITHER A QUADRATIC POLYNOMIAL OR A LINEAR FACTOR.  SOLUTION OF THE    *
C    * QUADRATIC IS DONE USING THE QUADRATIC FORMULA.                            *
C    *                                                                           *
C    *****************************************************************************
      COMPLEX A,DISC,ROOT,DUMMY,AAA
      COMPLEX CSQRT
      DIMENSION A(26),ROOT(25),MULTI(25)
      IF(NA.EQ.2) GO TO 7
      IF(NA.EQ.1) GO TO 5
      ROOT(NROOT+1)=0.0
      MULTI(NROOT+1)=1
      NROOT=NROOT+1
      GO TO 50
    5 ROOT(NROOT+1)=-A(2)/A(1)
      MULTI(NROOT+1)=1
      NROOT=NROOT+1
      GO TO 50
    7 DISC=A(2)*A(2)-(4.0*A(1)*A(3))
      BBB=CABS(DISC)
      IF(BBB.LT.EPST) GO TO 10
      DUMMY=CSQRT(DISC)
      AAA=2.0*A(1)
      ROOT(NROOT+1)=(-A(2)+DUMMY)/AAA
      ROOT(NROOT+2)=(-A(2)-DUMMY)/AAA
      MULTI(NROOT+1)=1
      MULTI(NROOT+2)=1
      NROOT=NROOT+2
      GO TO 50
   10 ROOT(NROOT+1)=(-A(2))/(2.0*A(1))
      MULTI(NROOT+1)=2
      NROOT=NROOT+1
   50 RETURN
      END

$ENTRY
```

TABLE VIII-B

DOUBLE PRECISION PROGRAM FOR NEWTON'S METHOD

```
$JOB 10414

C     ************************************************************************
C     *                                                                      *
C     * DOUBLE PRECISION PROGRAM FOR NEWTON'S METHOD                         *
C     *                                                                      *
C     *                                                                      *
C     * NEWTONS METHOD EXTRACTS THE ZEROS AND THEIR MULTIPLICITIES OF A      *
C     * POLYNOMIAL OF MAXIMUM DEGREE 25 BY COMPUTING A SEQUENCE OF APPROX-   *
C     * IMATIONS CONVERGING TO A ZERO OF THE POLYNOMIAL USING THE ITERATION  *
C     * FORMULA                                                              *
C     *              X(N+1) = X(N)-P(X(N))/P'(X(N)).                         *
C     *                                                                      *
C     ************************************************************************
        DOUBLE PRECISION RA,VA,RXZERO,VXZERO,RB,VB,RCOEF,VCOEF,RX,VX,RXINI
       1T,VXINIT,RC,VC,RD,VD,RPX,VPX,RDPX,VDPX,RXNEW,VXNEW,RXO,VXO,EPSCHK,
       2EPSCNV,EPSQ,EPSMUL,XSTART,XEND,ABPX,ABDPX
        DOUBLE PRECISION DSQRT
        DIMENSION RA(26),VA(26),RB(26),VB(26),RC(26),VC(26),RD(26),VD(26),
       1RCOEF(26),VCOEF(26),MULT(25),RXZERO(25),VXZERO(25),RX(25),VX(25),R
       2XINIT(25),VXINIT(25)
        COMMON EPSCHK,MAX,IO2
        IO1=5                                                                 112
        IO2=6                                                                 116
      1 READ(IO1,1000) NOPOLY,N,NIAP,MAX,EPSCNV,EPSQ,EPSMUL,XSTART,XEND,KC
       1HECK
        IF(KCHECK.EQ.1) STOP
        NA=N+1                                                                130
        READ(IO1,1010) (RA(I),VA(I),I=1,NA)                                   132
        WRITE(IO2,1030) NOPOLY,N
        WRITE(IO2,1040) (I,RA(I),VA(I),I=1,NA)
        WRITE(IO2,2060)
        WRITE(IO2,2000) NIAP
        WRITE(IO2,2010) MAX
        WRITE(IO2,2020) EPSCNV
        WRITE(IO2,2030) EPSMUL
        WRITE(IO2,2040) XSTART
        WRITE(IO2,2050) XEND
        IF(NIAP.NE.0) GO TO 3
        NIAP=N
        CALL GENAPP(RXZERO,VXZERO,NIAP,XSTART)
        GO TO 4
      3 READ(IO1,1020) (RXZERO(I),VXZERO(I),I=1,NIAP)
      4 NDEF=N
        L=1                                                                   152
        ITER=0                                                                154
        NROOT=0                                                               160
        IROOT=0
        ITIME=0
        ND=0
        IALTER=0                                                              164
        K=0                                                                   168
        RXO=RXZERO(L)                                                         180
        VXO=VXZERO(L)                                                         181
        DO 5 I=1,NA                                                           188
        RCOEF(I)=RA(I)                                                        190
      5 VCOEF(I)=VA(I)                                                        191
     10 CALL HORNER(RCOEF,VCOEF,RXO,VXO,NDEF,RB,VB,RC,VC,RPX,VPX,RDPX,VDPX
       1)
        ABPX=DSQRT(RPX*RPX+VPX*VPX)
```

TABLE VIII-B (Continued)

```
      ABDPX=DSQRT(RDPX*RDPX+VDPX*VDPX)
      IF(ABDPX.NE.0.0) GO TO 20
      IF(ABPX.EQ.0.0) GO TO 70
      GO TO 110
   20 CALL NEWTON(RPX,VPX,RDPX,VDPX,RXO,VXO,RXNEW,VXNEW)
      ITER=ITER+1                                                       200
      RXO=RXNEW                                                         204
      VXO=VXNEW                                                         205
      EPSCHK=EPSCNV
      CALL CHECK(RPX,VPX,RDPX,VDPX,RXO,VXO,KANS)
      IF(KANS.EQ.1) GO TO 70                                            194
      IF(ITER.GE.MAX) GO TO 40
      GO TO 10                                                          208
   40 CALL ALTER(RXZERO(L),VXZERO(L),IALTER,ITIME)
      IF(IALTER.GT.5) GO TO 110
      RXO=RXZERO(L)
      VXO=VXZERO(L)
      ITER=0                                                            244
      GO TO 10                                                          248
   60 ND=NDEF+1
      DO 65 J=1,ND
      RD(J)=RCOEF(J)
   65 VD(J)=VCOEF(J)
      GO TO 140
   70 NROOT=NROOT+1                                                     268
      K=K+1                                                             272
      MULT(K)=1                                                         276
      RX(K)=RXO                                                         280
      VX(K)=VXO                                                         281
      RXINIT(K)=RXZERO(L)                                               288
      VXINIT(K)=VXZERO(L)                                               289
      CALL HORNER(RCOEF,VCOEF,RXO,VXO,NDEF,RB,VB,RC,VC,RPX,VPX,RDPX,VDPX
     1)
   80 IF(NROOT.GE.N) GO TO 147
      NDEF=NDEF-1
      NUM=NDEF+1
      DO 105 I=1,NUM                                                    294
      RCOEF(I)=RB(I)                                                    296
  105 VCOEF(I)=VB(I)                                                    297
      CALL HORNER(RCOEF,VCOEF,RXO,VXO,NDEF,RB,VB,RC,VC,RPX,VPX,RDPX,VDPX
     1)
      ABPX=DSQRT(RPX*RPX+VPX*VPX)
      ABDPX=DSQRT(RDPX*RDPX+VDPX*VDPX)
      IF(ABDPX.NE.0.0) GO TO 107
      IF(ABPX.EQ.0.0) GO TO 130
      GO TO 110
  107 CONTINUE
      EPSCHK=EPSMUL
      CALL CHECK(RPX,VPX,RDPX,VDPX,RXO,VXO,KANS)
      IF(KANS.EQ.1) GO TO 130                                           300
  110 IF(NDEF.GT.2) GO TO 113
      IROOT=K
      CALL QUAD(RCOEF,VCOEF,NDEF,RX,VX,K,MULT,EPSQ)
      GO TO 150
  113 IF(L.LT.NIAP) GO TO 115
      IF(XEND.EQ.0.0) GO TO 60
      IF(XSTART.GT.XEND) GO TO 60
      NIAP=N
      CALL GENAPP(RXZERO,VXZERO,NIAP,XSTART)
      L=0
```

TABLE VIII-B (Continued)

```
  115 L=L+1
      RXO=RXZERO(L)                                                        312
      VXO=VXZERO(L)                                                        313
      ITER=0                                                               316
      IALTER=0                                                             320
      GO TO 10                                                             324
  130 MULT(K)=MULT(K)+1                                                    328
      NROOT=NROOT+1                                                        332
      GO TO 80                                                             336
  140 IF(K.EQ.0) GO TO 160                                                 338
  147 IROOT=K
  150 WRITE(IO2,1025)
      WRITE(IO2,1050)
      WRITE(IO2,1060) (I,RX(I),VX(I),MULT(I),RXINIT(I),VXINIT(I),I=1,IRO
     1OT)
      KKK=IROOT+1
      IF(IROOT.LT.K) WRITE(IO2,1062) (I,RX(I),VX(I),MULT(I),I=KKK,K)
      EPSCHK=EPSCNV
      CALL BETTER(K,RXZERO,VXZERO,RX,VX,NA,RA,VA,RCOEF,VCOEF,N,RC,VC,RB,
     1VB)
  160 IF(K.EQ.0) GO TO 170
      WRITE(IO2,1065)
      WRITE(IO2,1050)
      WRITE(IO2,1060) (I,RX(I),VX(I),MULT(I),RXINIT(I),VXINIT(I),I=1,IRO
     1OT)
      KKK=IROOT+1
      IF(IROOT.LT.K) WRITE(IO2,1062) (I,RX(I),VX(I),MULT(I),I=KKK,K)
  170 IF(ND.EQ.0) GO TO 1
      WRITE(IO2,1070)                                                      396
      WRITE(IO2,1075) (J,RD(J),VD(J),J=1,ND)
      GO TO 1                                                              444
 1000 FORMAT(3(I2,1X),9X,I3,8X,3(D6.0,1X),13X,2(D7.0,1X),I1)
 1010 FORMAT(2D30.0)
 1030 FORMAT(1H1,8X,43HNEWTONS METHOD TO FIND ZEROS OF POLYNOMIALS/9X,18
     1HPOLYNOMIAL NUMBER ,I2,11H OF DEGREE ,I2,////1X,28HTHE COEFFICIENT
     2S OF P(X) ARE/)
 1040 FORMAT(3X,2HP(,I2,4H) = ,D23.16,3H + ,D23.16,2H I)
 1020 FORMAT(2D30.0)
 1025 FORMAT(///1X,61HBEFORE THE ATTEMPT TO IMPROVE ACCURACY, THE ZEROS
     1OF P(X) ARE)
 1050 FORMAT(///2X,13HROOTS OF P(X),52X,14HMULTIPLICITIES,17X,21HINITIAL
     1 APPROXIMATION//)
 1060 FORMAT(3X,5HROOT(,I2,4H) = ,D23.16,3H + ,D23.16,2H I,7X,I2,7X,D23.
     116,3H + ,D23.16,2H I)
 1062 FORMAT(3X,5HROOT(,I2,4H) = ,D23.16,3H + ,D23.16,2H I,7X,I2,8X,23HS
     1OLVED BY DIRECT METHOD)
 1065 FORMAT(///1X,61HAFTER  THE ATTEMPT TO IMPROVE ACCURACY, THE ZEROS
     1OF P(X) ARE)
 1070 FORMAT(///1X,65HCOEFFICIENTS OF DEFLATED POLYNOMIAL FOR WHICH NO Z
     1EROS WERE FOUND/)
 1075 FORMAT(3X,2HD(,I2,4H) = ,D23.16,3H + ,D23.16,2H I)
 2000 FORMAT(1X,41HNUMBER OF INITIAL APPROXIMATIONS GIVEN.  ,I2)
 2010 FORMAT(1X,29HMAXIMUM NUMBER OF ITERATIONS.,11X,I3)
 2020 FORMAT(1X,21HTEST FOR CONVERGENCE.,13X,D9.2)
 2030 FORMAT(1X,24HTEST FOR MULTIPLICITIES.,10X,D9.2)
 2040 FORMAT(1X,23HRADIUS TO START SEARCH.,11X,D9.2)
 2050 FORMAT(1X,21HRADIUS TO END SEARCH.,13X,D9.2)
 2060 FORMAT(//1X)
      END                                                                  450
```

TABLE VIII-B (Continued)

```
      SUBROUTINE GENAPP(APPR,APPI,NAPP,XSTART)
C    ************************************************************************
C    *                                                                      *
C    * SUBROUTINE GENAPP GENERATES N INITIAL APPROXIMATIONS, WHERE N IS THE *
C    * DEGREE OF THE ORIGINAL POLYNOMIAL.                                   *
C    *                                                                      *
C    ************************************************************************
      DOUBLE PRECISION APPR,APPI,XSTART,DUMMY,BETA
      DOUBLE PRECISION DCOS,DSIN
      DIMENSION APPR(25),APPI(25)
      COMMON DUMMY,MAX,IO2
      IF(XSTART.EQ.0.0) XSTART=0.5
      BETA=0.2617994
      DO 10 I=1,NAPP
      APPR(I)=XSTART*DCOS(BETA)
      APPI(I)=XSTART*DSIN(BETA)
      BETA=BETA+0.5235988
   10 XSTART=XSTART+0.5
      RETURN
      END
```

```
      SUBROUTINE ALTER(XOLDR,XOLDI,NALTER,ITIME)
C    ************************************************************************
C    *                                                                      *
C    * SUBROUTINE ALTER ALTERS THE INITIAL APPROXIMATIONS WHICH PRODUCE NO  *
C    * CONVERGENCE TO A ZERO. THIS IS DONE A MAXIMUM OF 5 TIMES FOR EACH ROOT. *
C    *                                                                      *
C    ************************************************************************
      DOUBLE PRECISION XOLDR,XOLDI,DUMMY,ABXOLD,BETA
      DOUBLE PRECISION DCOS,DSIN,DATAN2
      DOUBLE PRECISION DSQRT
      COMMON DUMMY,MAX,IO2
      IF(ITIME.NE.0) GO TO 5
      ITIME =1
      WRITE(IO2,1010) MAX
    5 IF(NALTER.EQ.0) GO TO 10
      WRITE(IO2,1000) XOLDR,XOLDI
      GO TO 20
   10 ABXOLD=DSQRT(XOLDR*XOLDR+XOLDI*XOLDI)
      BETA=DATAN2(XOLDI,XOLDR)
      WRITE(IO2,1020) XOLDR,XOLDI
   20 NALTER=NALTER+1
      IF(NALTER.GT.5) RETURN
      GO TO (30,40,30,40,30),NALTER
   30 XOLDR=-XOLDR
      XOLDI=-XOLDI
      GO TO 50
   40 BETA=BETA+1.0471976
      XOLDR=ABXOLD*DCOS(BETA)
      XOLDI=ABXOLD*DSIN(BETA)
   50 RETURN
 1000 FORMAT(1X,D23.16,3H + ,D23.16,2H I,10X,21HALTERED APPROXIMATION)
 1010 FORMAT(///1X,54HNO CONVERGENCE FOR THE FOLLOWING APPROXIMATIONS AF
     1TER ,I3,12H ITERATIONS.//)
 1020 FORMAT(/1X,D23.16,3H + ,D23.16,2H I,10X,21HINITIAL APPROXIMATION)
      END
```

TABLE VIII-B (Continued)

```
      SUBROUTINE QUAD(UA,VA,NA,UROOT,VROOT,NROOT,MULTI,EPST)
C    **************************************************************************
C    *                                                                        *
C    * SUBROUTINE QUAD SOLVES DIRECTLY FOR THE ZEROS AND THEIR MULTIPLICITIES *
C    * OF EITHER A QUADRATIC POLYNOMIAL OR A LINEAR FACTOR.  SOLUTION OF THE  *
C    * QUADRATIC IS DONE USING THE QUADRATIC FORMULA.                         *
C    *                                                                        *
C    **************************************************************************
      DOUBLE PRECISION UA,VA,UROOT,VROOT,BBB,UAAA,VAAA,UDISC,VDISC,UDUMM
     1Y,VDUMMY,RDUMMY,SDUMMY,EPST,UBBB,VBBB
      DOUBLE PRECISION DSQRT
      DIMENSION UA(26),VA(26),UROOT(25),VROOT(25),MULTI(25)
      IF(NA.EQ.2) GO TO 7
      IF(NA.EQ.1) GO TO 5
      UROOT(NROOT+1)=0.0
      VROOT(NROOT+1)=0.0
      MULTI(NROOT+1)=1
      NROOT=NROOT+1
      GO TO 50
    5 BBB=UA(1)*UA(1)+VA(1)*VA(1)
      UROOT(NROOT+1)=(-UA(2)*UA(1)-VA(2)*VA(1))/BBB
      VROOT(NROOT+1)=(-VA(2)*UA(1)+UA(2)*VA(1))/BBB
      MULTI(NROOT+1)=1
      NROOT=NROOT+1
      GO TO 50
    7 UDISC=(UA(2)*UA(2)-VA(2)*VA(2))-(4.0*(UA(1)*UA(3)-VA(1)*VA(3)))
      VDISC=(VA(2)*UA(2)+UA(2)*VA(2))-(4.0*(VA(1)*UA(3)+UA(1)*VA(3)))
      BBB=DSQRT(UDISC*UDISC+VDISC*VDISC)
      IF(BBB.LT.EPST) GO TO 10
      CALL COMSQT(UDISC,VDISC,UDUMMY,VDUMMY)
      UBBB=-UA(2)+UDUMMY
      VBBB=-VA(2)+VDUMMY
      RDUMMY=-UA(2)-UDUMMY
      SDUMMY=-VA(2)-VDUMMY
      UAAA=2.0*UA(1)
      VAAA=2.0*VA(1)
      BBB=UAAA*UAAA+VAAA*VAAA
      UROOT(NROOT+1)=(UBBB*UAAA+VBBB*VAAA)/BBB
      VROOT(NROOT+1)=(VBBB*UAAA-UBBB*VAAA)/BBB
      UROOT(NROOT+2)=(RDUMMY*UAAA+SDUMMY*VAAA)/BBB
      VROOT(NROOT+2)=(SDUMMY*UAAA-RDUMMY*VAAA)/BBB
      MULTI(NROOT+1)=1
      MULTI(NROOT+2)=1
      NROOT=NROOT+2
      GO TO 50
   10 UAAA=2.0*UA(1)
      VAAA=2.0*VA(1)
      BBB=UAAA*UAAA+VAAA*VAAA
      UROOT(NROOT+1)=(-UA(2)*UAAA-VA(2)*VAAA)/BBB
      VROOT(NROOT+1)=(-VA(2)*UAAA+UA(2)*VAAA)/BBB
      MULTI(NROOT+1)=2
      NROOT=NROOT+1
   50 RETURN
      END
```

TABLE VIII-B (Continued)

```
      SUBROUTINE COMSQT(UX,VX,UY,VY)
C     *************************************************************************
C     *                                                                       *
C     * THIS SUBROUTINE COMPUTES THE SQUARE ROOT OF A COMPLEX NUMBER.         *
C     *                                                                       *
C     *************************************************************************
      DOUBLE PRECISION UX,VX,UY,VY,DUMMY,R,AAA,BBB
      DOUBLE PRECISION DSQRT,DABS
      R=DSQRT(UX*UX+VX*VX)
      AAA=DSQRT(DABS((R+UX)/2.0))
      BBB=DSQRT(DABS((R-UX)/2.0))
      IF(VX) 10,20,30
   10 UY=AAA
      VY=-1.0*BBB
      GO TO 100
   20 IF(UX) 40,50,60
   30 UY=AAA
      VY=BBB
      GO TO 100
   40 DUMMY=DABS(UX)
      UY=0.0
      VY=DSQRT(DUMMY)
      GO TO 100
   50 UY=0.0
      VY=0.0
      GO TO 100
   60 DUMMY=DABS(UX)
      UY=DSQRT(DUMMY)
      VY=0.0
  100 RETURN
      END


      SUBROUTINE HORNER(RA,VA,RXO,VXO,NDEF,RB,VB,RC,VC,RPX,VPX,RDPX,VDPX
C     *************************************************************************
C     *                                                                       *
C     * HORNER'S METHOD COMPUTES THE VALUE OF A POLYNOMIAL P(X) AT A POINT D AND *
C     * ITS DERIVATIVE AT D.  SYNTHETIC DIVISION IS USED TO DEFLATE THE       *
C     * POLYNOMIAL BY DIVIDING OUT THE FACTOR (X-D).                          *
C     *                                                                       *
C     *************************************************************************
     1)
      DOUBLE PRECISION VDPX,RXO,VXO,RB,VB,RC,VC,RPX,VPX,RDPX,RA,VA
      DIMENSION RA(26),VA(26),RB(26),VB(26),RC(26),VC(26)
      RB(1)=RA(1)                                                      516
      VB(1)=VA(1)                                                      517
      NUM=NDEF+1                                                       520
      DO 10 I=2,NUM                                                    524
      RB(I)=RA(I)+(RB(I-1)*RXO-VB(I-1)*VXO)
   10 VB(I)=VA(I)+(VB(I-1)*RXO+RB(I-1)*VXO)
      RPX=RB(NUM)                                                      532
      VPX=VB(NUM)                                                      533
      RC(1)=RB(1)                                                      540
      VC(1)=VB(1)                                                      541
      IF(NDEF.LT.2) GO TO 25
      DO 20 J=2,NDEF                                                   544
      RC(J)=RB(J)+(RC(J-1)*RXO-VC(J-1)*VXO)
   20 VC(J)=VB(J)+(VC(J-1)*RXO+RC(J-1)*VXO)
   25 RDPX=RC(NDEF)
      VDPX=VC(NDEF)                                                    553
      RETURN                                                           572
      END                                                              580
```

TABLE VIII-B (Continued)

```
      SUBROUTINE NEWTON(RPX,VPX,RDPX,VDPX,RXO,VXO,RXNEW,VXNEW)                  600
C     *************************************************************************
C     *                                                                       *
C     * THIS SUBROUTINE CALCULATES A NEW APPROXIMATION FROM THE OLD APPROX-   *
C     * IMATION BY USING THE ITERATION FORMULA                                *
C     *              X(N+1) = X(N)-P(X(N))/P'(X(N)).                          *
C     *                                                                       *
C     *************************************************************************
      DOUBLE PRECISION RPX,VPX,RDPX,VDPX,RXO,VXO,RXNEW,VXNEW,ARG
      DOUBLE PRECISION DSQRT
      DOUBLE PRECISION DDD
      ARG=RDPX*RDPX+VDPX*VDPX
      DDD=DSQRT(ARG)
      IF(DDD.EQ.0.0) RETURN
      RXNEW=RXO-((RPX*RDPX+VPX*VDPX)/ARG)
      VXNEW=VXO-((VPX*RDPX-RPX*VDPX)/ARG)
      RETURN                                                                    616
      END                                                                       620
```

```
      SUBROUTINE CHECK(RPX,VPX,RDPX,VDPX,RXO,VXO,KANS)
C     *************************************************************************
C     *                                                                       *
C     * THIS SUBROUTINE CHECKS FOR CONVERGENCE OF THE SEQUENCE OF APPROX-     *
C     * IMATIONS BY TESTING THE EXPRESSION                                    *
C     * ABSOLUTE VALUE OF (P(X(N))/P'(X(N)))/ABSOLUTE VALUE OF X(N+1).        *
C     * WHEN IT IS AS SMALL AS DESIRED, CONVERGENCE IS OBTAINED.              *
C     *                                                                       *
C     *************************************************************************
      DOUBLE PRECISION RPX,VPX,RDPX,VDPX,RXO,VXO,ABSXO,ABSQUO,RDUMMY,VDU    749
     1MMY,EPS                                                                750
      DOUBLE PRECISION ARG
      DOUBLE PRECISION DSQRT
      DOUBLE PRECISION DDD
      COMMON EPS,MAX,IO2
      ABSXO=DSQRT(RXO*RXO+VXO*VXO)
      IF(ABSXO.EQ.0.) GO TO 25
      ARG=RDPX*RDPX+VDPX*VDPX
      DDD=DSQRT(ARG)
      IF(DDD.EQ.0.0) GO TO 25
      RDUMMY=(RPX*RDPX+VPX*VDPX)/ARG
      VDUMMY=(VPX*RDPX-RPX*VDPX)/ARG
      ABSQUO=DSQRT(RDUMMY*RDUMMY+VDUMMY*VDUMMY)
      IF(ABSQUO/ABSXO.LT.EPS) GO TO 10
      KANS=0                                                                    760
      RETURN                                                                    764
   10 KANS=1                                                                    768
      RETURN                                                                    772
   25 KANS=0
      RETURN
      END                                                                       780
```

TABLE VIII-B (Continued)

```
      SUBROUTINE BETTER(K,RXZERO,VXZERO,RX,VX,NA,RA,VA,RCOEF,VCOEF,N,RC,
     1VC,RB,VB)
C     ************************************************************************
C     *                                                                      *
C     * SUBROUTINE BETTER ATTEMPTS TO IMPROVE THE ACCURACY OF THE ZEROS FOUND *
C     * BY USING THEM AS INITIAL APPROXIMATIONS WITH NEWTON'S METHOD APPLIED TO *
C     * THE FULL, UNDEFLATED POLYNOMIAL.                                     *
C     *                                                                      *
C     ************************************************************************
      DOUBLE PRECISION RXZERO,VXZERO,RX,VX,RA,VA,RCOEF,VCOEF,RC,VC,RB,VB      805
     1,RXO,VXO,RPX,VPX,RDPX,VDPX,RXNEW,VXNEW,EPS
      DOUBLE PRECISION DSQRT
      DIMENSION RXZERO(25),VXZERO(25),RX(25),VX(25),RA(26),VA(26),RCOEF(
     126),VCOEF(26),RC(26),VC(26),RB(26),VB(26)
      DOUBLE PRECISION ABPX,ABDPX
      COMMON EPS,MAX,IO2
      DO 10 I=1,K                                                             812
      RXZERO(I)=RX(I)                                                         815
   10 VXZERO(I)=VX(I)                                                         816
      DO 20 I=1,NA
      RCOEF(I)=RA(I)                                                          824
   20 VCOEF(I)=VA(I)                                                          825
      DO 50 J=1,K                                                             828
      RXO=RXZERO(J)                                                           832
      VXO=VXZERO(J)                                                           833
      NN=N                                                                    834
      ITER=0                                                                  836
   30 CALL HORNER(RCOEF,VCOEF,RXO,VXO,NN,RB,VB,RC,VC,RPX,VPX,RDPX,VDPX)
      ABPX=DSQRT(RPX*RPX+VPX*VPX)
      ABDPX=DSQRT(RDPX*RDPX+VDPX*VDPX)
      IF(ABDPX.NE.0.0) GO TO 33
      IF(ABPX.EQ.0.0) GO TO 40
      GO TO 34
   33 CALL NEWTON(RPX,VPX,RDPX,VDPX,RXO,VXO,RXNEW,VXNEW)
      ITER=ITER+1                                                             856
      RXO=RXNEW                                                               860
      VXO=VXNEW                                                               861
      CALL CHECK(RPX,VPX,RDPX,VDPX,RXO,VXO,KANS)
      IF(KANS.EQ.1) GO TO 40                                                  844
      IF(ITER.GE.MAX) GO TO 35
      GO TO 30                                                                864
   34 WRITE(IO2,1112) RXO,VXO
   35 WRITE(IO2,100) J,RXZERO(J),VXZERO(J)
      WRITE(IO2,200) MAX
   40 RX(J)=RXO                                                               870
      VX(J)=VXO                                                               871
   50 CONTINUE                                                                872
      RETURN                                                                  876
 1112 FORMAT(1HO,36HTHE VALUE OF THE DERIVATIVE AT XO = ,D23.16,3H + ,D2
     13.16,2H I,10H  IS ZERO.)
  100 FORMAT(42HOIN THE ATTEMPT TO IMPROVE ACCURACY, ROOT(,I2,4H) = ,D23
     1.16,3H + ,D23.16,2H I,18H DID NOT CONVERGE.)
  200 FORMAT(33H THE PRESENT APPROXIMATION AFTER ,I3,29H ITERATIONS IS P
     1RINTED BELOW.)
      END                                                                     880

$ENTRY
```

APPENDIX C

MULLER'S METHOD

1. Use of the Programs

Two programs using Muller's method are presented here. The first is the single precision program. The second program is in double precision and is desigend to perform double precision complex arithmetic. These programs are written for use on any computer using FORTRAN IV language. They have been tested on the IBM S/360 mod. 50 computer which has a 32 bit word. However, it may be necessary to change the system functions as described below. The single precision program may be changed to double precision as described below.

After selecting the desired program, the input data should be prepared as described in section 2.

Each program is designed to solve polynomials of degree 25 or less. Both the coefficient of the highest degree term and the constant coefficient should be non-zero. In order to solve polynomials of degree N, where N > 25, certain array dimensions must be changed. These are listed in Table IX for the main program and subprograms in both single precision and double precision.

TABLE IX

PROGRAM CHANGES FOR SOLVING POLYNOMIALS OF DEGREE GREATER THAN 25 BY MULLER'S METHOD

Single Precision	Double Precision
Main Program	
ROOT(N)	UROOT(N),VROOT(N)
MULT(N)	MULT(N)
APP(N,3)	UAPP(N,3),VAPP(N,3)
WORK(N+1)	UWORK(N+1),VWORK(N+1)
B(N+1)	UB(N+1),VB(N+1)
A(N+1)	UA(N+1),VA(N+1)
RAPP(N,3)	URAPP(N,3),VRAPP(N,3)
Subroutine BETTER	
ROOT(N)	UROOT(N),VROOT(N)
A(N+1)	UA(N+1),VA(N+1)
BAPP(N,3)	UBAPP(N,3),VBAPP(N,3)
B(N+1)	UB(N+1),VB(N+1)
ROOTS(N)	UROOTS(N),VROOTS(N)
RAPP(N,3)	URAPP(N,3),VRAPP(N,3)
MULT(N)	MULT(N)
Subroutine GENAPP	
APP(N,3)	APPR(N,3),APPI(N,3)
Subroutine HORNER	
A(N+1)	UA(N+1),VA(N+1)
B(N+1)	UB(N+1),VB(N+1)
Subroutine QUAD	
A(N+1)	UA(N+1),VA(N+1)
ROOT(N)	UROOT(N),VROOT(N)
MULTI(N)	MULTI(N)

Certain computers may require that the system functions of Table X be changed in the single precision and double precision programs. A "c" denotes a complex number and an "r" denotes a real number.

TABLE X

SYSTEM FUNCTIONS USED IN MULLER'S METHOD

Single Precision		Double Precision
CABS(c)	- obtain absolute value -	DABS(r)
AIMAG(c)	- obtain imaginary part	
REAL(c)	- obtain real part	
ATAN2(r_1,r_2)	- arctangent of r_1/r_2 -	DATAN2(r_1,r_2)
SQRT(r)	- square root -	DSQRT(r)
CMPLX(r_1,r_2)	- express two real numbers in complex form	
COS(r)	- cosine of angle -	DCOS(r)
SIN(r)	- sine of angle -	DSIN(r)
CSQRT(c)	- square root -	DSQRT(r)

When used on the IBM S/360 with the WATFOR complier for FORTRAN IV, the system functions in Table X-A must be typed in a declaration statement. These also appear in the program listing. For use without the WATFOR compiler or on other machines, these system functions might have to be removed. A "c" denotes a complex number and an "r" denotes a real number.

TABLE X-A

SYSTEM FUNCTIONS IN MULLER'S METHOD TO BE TYPED WHEN THE WATFOR COMPILER IS USED

Single Precision		Double Precision
	Main Program and Subroutine TEST	
	square root -	DSQRT(r)
	Subroutine CALC	
CMPLX(r_1,r_2)	- express in complex form	
	arctangent of r_1/r_2 -	DATAN2(r_1,r_2)
	cosine of angle -	DCOS(r)
	sine of angle -	DSIN(r)
	square root -	DSQRT(r)
	Subroutine GENAPP	
CMPLX(r_1,r_2)	- express in complex form	
	cosine of angle -	DCOS(r)
	sine of angle -	DSIN(r)
	Subroutine ALTER	
CMPLX(r_1,r_2)	- express in complex form	
	cosine of angle -	DCOS(r)
	sine of angle -	DSIN(r)
	square root -	DSQRT(r)
	arctangent of r_1/r_2 -	DATAN2(r_1,r_2)
	Subroutine QUAD	
CSQRT(c)	- square root -	DSQRT(r)
	Subroutine COMSQT	
	square root -	DSQRT(r)
	absolute value -	DABS(r)

The single precision program may be converted to double precision for use on machines equipped to perform double precision complex arithmetic provided the following changes or their equivalent are made and the system functions of Table XI are used and typed in a declaration statement where necessary. The changes presented below are those required for the IBM S/360. A "c" denotes a comples number and an "r" denotes a real number. The format statements should be changed from E-type to D-type.

In the main program and each subprogram change COMPLEX $c_1, c_2, \ldots$ to COMPLEX*16 $c_1, c_2, \ldots$ and add IMPLICIT REAL*8(A-H,O-Z).

TABLE XI

SYSTEM FUNCTIONS FOR CONVERTING SINGLE PRECISION MULLER'S METHOD TO DOUBLE PRECISION

Single Precision	changed to	Double Precision
	Main Program and Subroutines TEST and QUAD	
CABS(c)	- absolute value -	CDABS(c)
	Subroutine CALC	
	Add	COMPLEX*8 SISC
CABS(c)	- absolute value -	CDABS(c)
AIMAG(c)	- obtain imaginary part -	AIMAG(c)(single precision)
REAL(c)	- obtain real part -	REAL(c)(single precision)
ATAN2(r_1,r_2)	- arctangent of r_1/r_2 -	DATAN2(r_1,r_2)
SQRT(r)	- square root -	DSQRT(r)
CMPLX(r_1,r_2)	- express in complex form -	DCMPLX(r_1,r_2)
COS(r)	- cosine of angle -	DCOS(r)
SIN(r)	- sine of angle -	DSIN(r)
	Subroutine GENAPP	
COS(r)	- cosine of angle -	DCOS(r)
SIN(r)	- sine of angle -	DSIN(r)
CMPLX(r_1,r_2)	- express in complex form -	DCMPLX(r_1,r_2)
	Subroutine ALTER	
	Add	COMPLEX*8 SX2
AIMAG(c)	- obtain imaginary part -	AIMAG(c)(single precision)

TABLE XI (Continued)

Single Precision	changed to	Double Precision
REAL(c)	- obtain real part -	REAL(c)(single precision)
CABS(c)	- absolute value -	CDABS(c)
ATAN2(r_1,r_2)	- arctangent of r_1/r_2 -	DATAN2(r_1,r_2)
COS(r)	- cosine of angle -	DCOS(r)
SIN(r)	- sine of angle -	DSIN(r)
CMPLX(r_1,r_2)	- express in complex form-	DCMPLX(r_1,r_2)
	Subroutine QUAD	
CSQRT(c)	- square root -	CDSQRT(c)

2. Input Data for Muller's Method

The input data for Muller's method is identical to the input data for Newton's method as described in Appendix B, § 2 except for the variable names. The correspondence of input variable names is given in Table XII. Only one (not three) initial approximation, X_0, is given for each root. The other two required by Muller's method are constructed within the program and are $.9X_0$ and $1.1X_0$.

3. Variables Used in Muller's Method

The definitions of the major variables used in Muller's method are given in Table XIII. For definitions of variables not listed in this table see the definitions of variables for the corresponding subroutine in Table VIII of Appendix B. The notation and symbols used here are the same as for table VIII and are described in Appendix B, § 3.

TABLE XII

CORRESPONDENCE OF NEWTON'S AND MULLER'S INPUT DATA VARIABLES

Newton's Method	Muller's Method
Control Card	
NOPOLY	NOPOLY
N	NP
NIAP	NAPP
MAX	MAX
EPSCNV	EPS
EPSQ	EPSO
EPSMUL	EPSM
XSTART	XSTART
XEND	XEND
KCHECK	KCHECK
Coefficient Card	
A (RA)	A (UA)
A (VA)	A (VA)
Initial Approximation Card	
XZERO (RXZERO)	APP (UAPP)
XZERO (VXZERO)	APP (VAPP)
End Card	
KCHECK	KCHECK

4. Description of Program Output

The output from Muller's method is the same as that for Newton's method as described in Appendix B, § 4. Only one initial approximation, Z, (not three) is printed for each root. It is either that supplied by the user or generated by the program. The other two approximations used were 0.9Z and 1.1Z.

5. Informative and Error Messages

The output may contain informative messages printed as an aid to the user. These are:

"NO ZEROS WERE FOUND FOR POLYNOMIAL NUMBER XX."

XX is the number of the polynomial. This message is printed if no roots of the polynomial were found.

```
"IN THE ATTEMPT TO IMPROVE ACCURACY, ROOT(X) = YYY
    DID NOT CONVERGE AFTER ZZZ ITERATIONS
      THE PRESENT APPROXIMATION IS AAA"
```

X is the number of the root before the attempt to improve accuracy, YYY is the value of the root before attempt to improve accuracy, ZZZ is the maximum number of iterations, and AAA is the current approximation after the maximum number of iterations. This message has the same meaning as the corresponding message in Appendix B, § 5.

TABLE XIII

VARIABLES USED IN MULLER'S METHOD

Single Precision Variable	Type	Double Precision Variable	Type	Disposition of Argument	Description
					Main Program
NP	I	NP	I		Degree of polynomial P(X)
NROOT	I	NROOT	I		Number of distinct roots found
NOMULT	I	NOMULT	I		Number of roots (counting multiplicities)
ROOT	C	UROOT,VROOT	R		Array containing the roots
NAPP	I	NAPP	I		Number of initial approximations to be read in
APP	C	UAPP,VAPP	R		Array of initial approximations
WORK	C	UWORK,VWORK	R		Working array containing coefficients of current polynomial
B	C	UB,VB	R		Array containing coefficients of deflated polynomial
A	C	UA,VA	R		Array containing coefficients of original polynomial, P(X)
RAPP	C	URAPP,VRAPP	R		Array of initial or altered approximations for which convergence was obtained
X1	C	UX1,VX1	R		One of three current approximations to a root
X2	C	UX2,VX2	R		One of three current approximations to a root
X3	C	UX3,VX3	R		One of three current approximations to a root
PX1	C	UPX1,VPX1	R		Value of polynomial P(X) at X1
PX2	C	UPX2,VPX2	R		Value of polynomial P(X) at X2
PX3	C	UPX3,VPX3	R		Value of polynomial P(X) at X3
X4	C	UX4,VX4	R		Newest approximation (X_{n+1}) to root
PX4	C	UPX4,VPX4	R		Value of polynomial P(X) at X4
MULT	I	MULT	I		Array containing the multiplicities of each root found
ITER	I	ITER	I		Counter for iterations
IO1	I	IO1	I		Unit number of input device
IO2	I	IO2	I		Unit number of output device

TABLE XIII (Continued)

Single Precision		Double Precision		Disposition	
Variable	Type	Variable	Type	of Argument	Description
EPSRT	R	EPSRT	R		Number used in subroutine BETTER to generate two approximations from the one given
NOPOLY	I	NOPOLY	I		Number of the polynomial
MAX	I	MAX	I		Maximum number of iterations
EPS	R	EPS	R		Tolerance check for convergence
EPSO	R	EPSO	R		Tolerance check for zero (0)
EPSM	R	EPSM	R		Tolerance check for multiplicities
KCHECK	I	KCHECK	I		Program control, KCHECK = 1 stops execution of program
XSTART	R	XSTART	R		Magnitude at which to start generating initial approximations
XEND	R	XEND	R		Magnitude at which to end generating initial approximations
NWORK	I	NWORK	I		Degree of current deflated polynomial whose coefficients are in WORK
ITIME	I	ITIME	I		Program control
NALTER	I	NALTER	I		Number of alterations which have been performed on an initial approximation
IAPP	I	IAPP	I		Counter for number of initial approximations used
CONV	L	CONV	L		When CONV is true, convergence has been obtained
IROOT	I	IROOT	I		Number of distinct roots solved by Muller's method, i.e. not solved directly by subroutine QUAD
					Subroutine HORNER
A	C	UA,VA	R	E	Array of current polynomial coefficients (to be deflated or evaluated)
NA	I	NA	I	E	Degree of polynomial to be deflated or evaluated
X	C	UX,VX	R	E	Approximation at which to evaluate or deflate the polynomial

TABLE XIII (Continued)

Single Precision Variable	Type	Double Precision Variable	Type	Disposition of Argument	Description
B	C	UB,VB	R	R	Array containing the coefficients of the deflated polynomial
PX	C	UPX,VPX	R	R	Value of the polynomial at X
NUM	I	NUM	I		Number of coefficients of polynomial to be deflated
					Subroutine TEST
X3	C	UX3,VX3	R	E	Approximation to Root (old) (X_n)
X4	C	UX4,VX4	R	E	New approximation to root (X_{n+1})
CONV	L	CONV	L	R	CONV = true implies convergence has been obtained
EPS	R	EPS	R	C	Tolerance for convergence test
EPSO	R	EPSO	R	C	Tolerance check for zero (0)
DENOM	R	DENOM	R		Magnitude of new approximation, (X_{n+1})
					Subroutine BETTER
MULT	I	MULT	I	ECR	Array of multiplicities of each root
A	C	UA,VA	R	E	Array of coefficients of original undeflated polynomial
NP	I	NP	I	E	Degree of original polynomial
ROOT	C	UROOT,VROOT	R	ECR	Array of roots
NROOT	I	NROOT	I	ECR	Number of roots stored in root
BAPP	C	UBAPP,VBAPP	R	E	Array of initial approximations (old roots)
IROOT	I	IROOT	I	ECR	Number of roots solved by the iterative process (Not QUAD)
ROOTS	C	UROOTS,VROOTS	R		Temporary storage for new (better) roots
L	I	L	I		Number of roots found by BETTER
EPSRT	R	EPSRT	R	C	A small number used to generate two of the three approximations when given one
ITER	I	ITER	I	C	Counter for number of iterations

TABLE XIII (Continued)

Single Precision Variable	Type	Double Precision Variable	Type	Disposition of Argument	Description
B	C	UB,VB	R		Array containing coefficients of deflated polynomial
X1	C	UX1,VX1	R		One of three approximations to the root
X2	C	UX2,VX2	R		One of three approximations to the root
X3	C	UX3,VX3	R		One of three approximations to the root
PX1	C	UPX1,VPX1	R		Value of polynomial (P(X)) at X1
PX2	C	UPX2,VPX2	R		Value of polynomial (P(X)) at X2
PX3	C	UPX3,VPX3	R		Value of polynomial (P(X)) at X3
CONV	L	CONV	L		CONV = true implies convergence has been obtained
X4	C	UX4,VX4	R		Newest approximation to root
J	I	J	I		Program control - counts the number of roots used as initial approximations
MAX	I	MAX	I	C	Maximum number of iterations permitted
IO2	I	IO2	I	C	Unit number of output device

Subroutine ALTER

X1	C	X1R,X1I	R	ECR	One of the three approximations to be altered
X2	C	X2R,X2I	R	ECR	One of the three approximations to be altered
X3	C	X3R,X3I	R	ECR	One of the three approximations to be altered
X2R	R	X2R	R		Real part of complex approximation
X2I	R	X2I	R		Imaginary part of complex approximation

Subroutine QUAD

EPST	R	EPST	R	E	Tolerance check for zero (0)

Subroutine CALC

These variables are dummy variables used for temporary storage and thus, are not defined.

MAIN PROGRAM

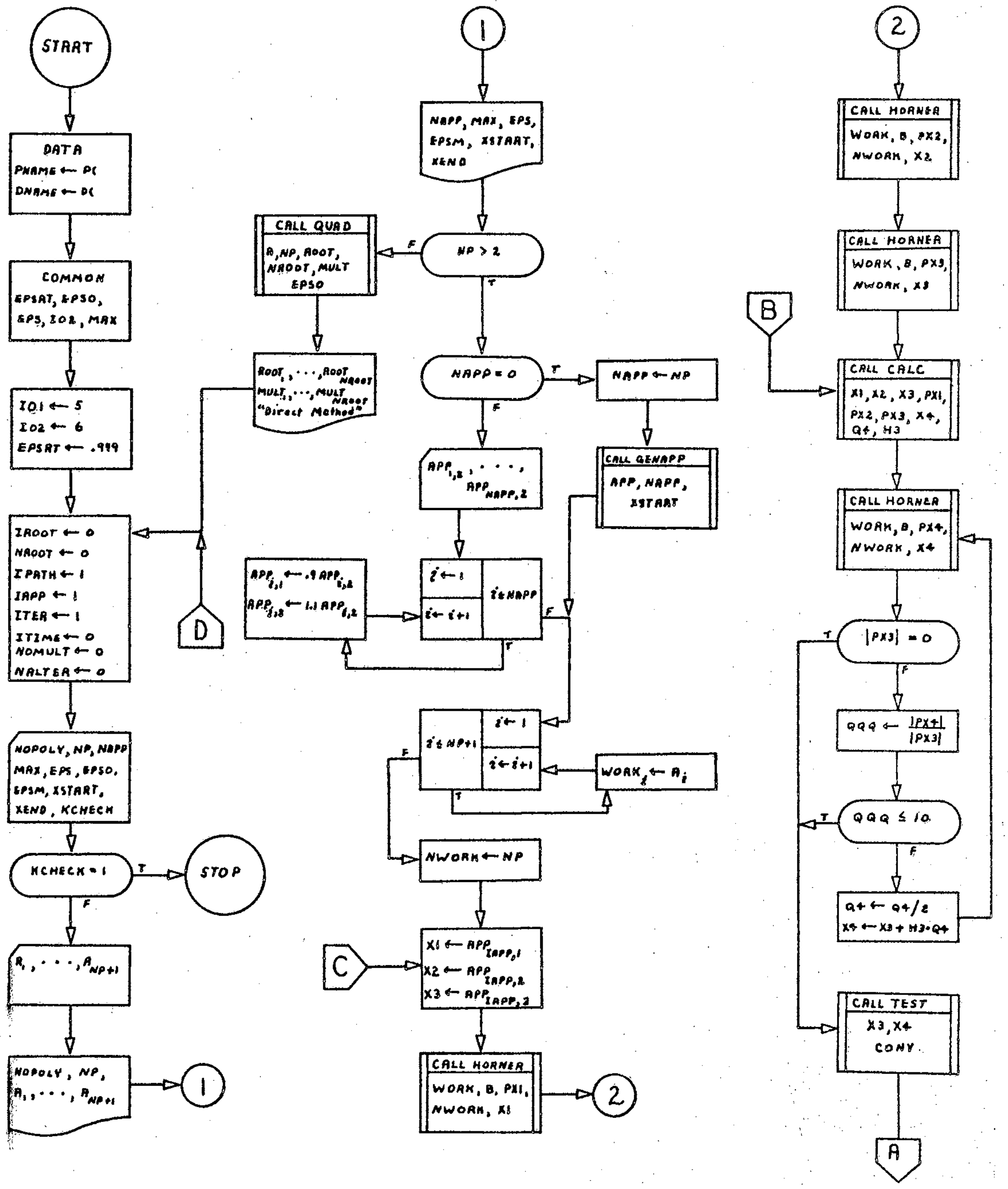


Figure 12.1. Flow Charts for Muller's Method

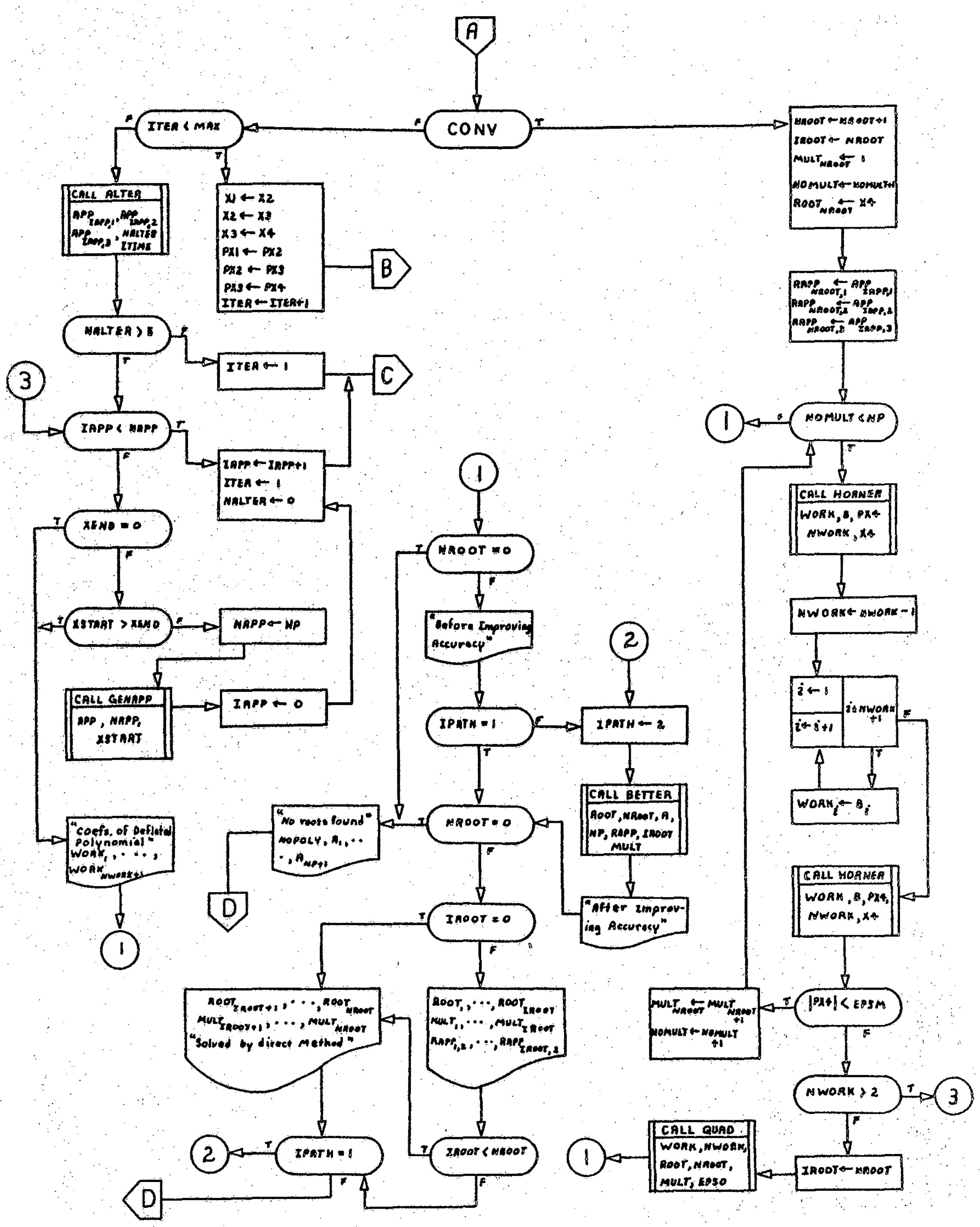
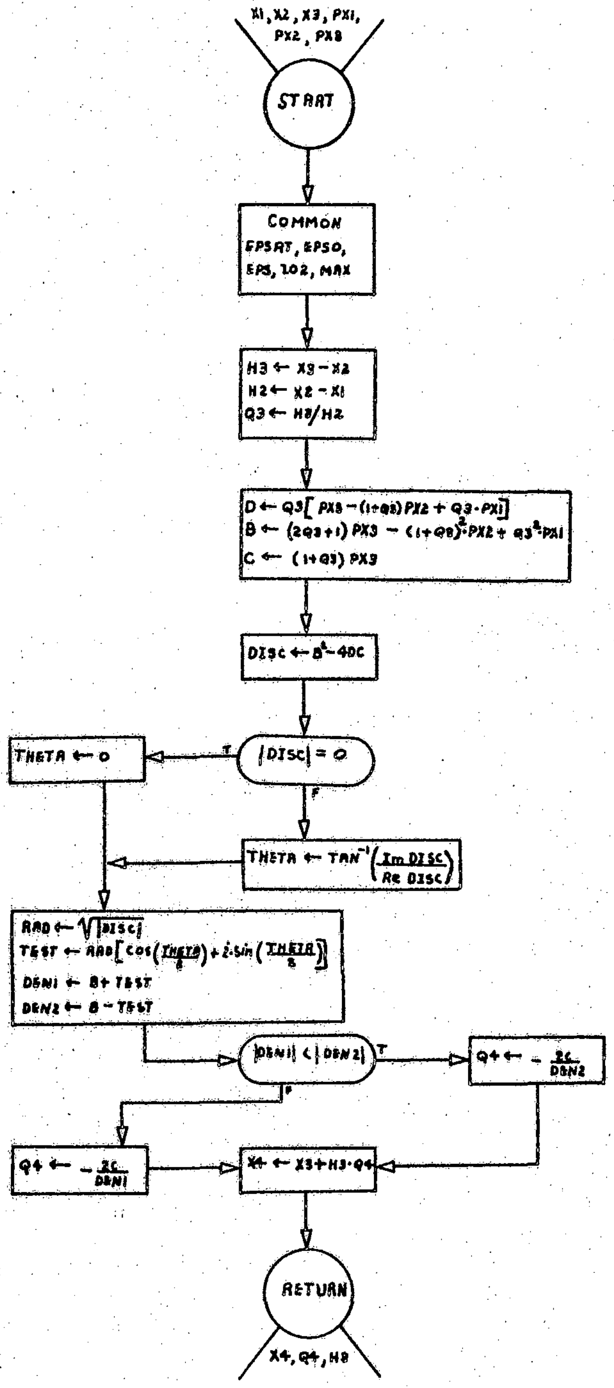

A
CONV
ITER < MAX
CALL ALTER
APP_{IAPP,1}, APP_{IAPP,2}, APP_{IAPP,3}, NALTER, ITIME
X1 ← X2
X2 ← X3
X3 ← X4
PX1 ← PX2
PX2 ← PX3
PX3 ← PX4
ITER ← ITER+1
B
NALTER > 5
ITER ← 1
C
3
IAPP < NAPP
IAPP ← IAPP+1
ITER ← 1
NALTER ← 0
XEND = 0
XSTART > XEND
NAPP ← NP
CALL GENAPP
APP, NAPP, XSTART
IAPP ← 0
"Coefs. of Deflated Polynomial" WORK_1, ..., WORK_{NWORK+1}
1
NROOT ← NROOT+1
IROOT ← NROOT
MULT_{NROOT} ← 1
NOMULT ← NOMULT+1
ROOT_{NROOT} ← X4
RAPP_{NROOT,1} ← APP_{IAPP,1}
RAPP_{NROOT,2} ← APP_{IAPP,2}
RAPP_{NROOT,3} ← APP_{IAPP,3}
NOMULT < NP
CALL HORNER
WORK, B, PX4, NWORK, X4
NWORK ← NWORK - 1
i ← 1
i ← i+1
i ≤ NWORK+1
WORK_i ← B_i
CALL HORNER
WORK, B, PX4, NWORK, X4
|PX4| < EPSM
MULT_{NROOT} ← MULT_{NROOT}+1
NOMULT ← NOMULT+1
NWORK > 2
IROOT ← NROOT
CALL QUAD
WORK, NWORK, ROOT, NROOT, MULT, EPSO
NROOT = 0
"Before Improving Accuracy"
IPATH = 1
2
IPATH ← 2
CALL BETTER
ROOT, NROOT, A, NP, RAPP, IROOT, MULT
"After Improving Accuracy"
NROOT = 0
"No roots found" NOPOLY, A_1, ..., A_{NP+1}
D
IROOT = 0
ROOT_{IROOT+1}, ..., ROOT_{NROOT}
MULT_{IROOT+1}, ..., MULT_{NROOT}
"Solved by direct Method"
ROOT_1, ..., ROOT_{IROOT}
MULT_1, ..., MULT_{IROOT}
RAPP_{1,2}, ..., RAPP_{IROOT,2}
IPATH = 1
IROOT < NROOT
T
F

Figure 12.1. (Continued)

CALC



TEST

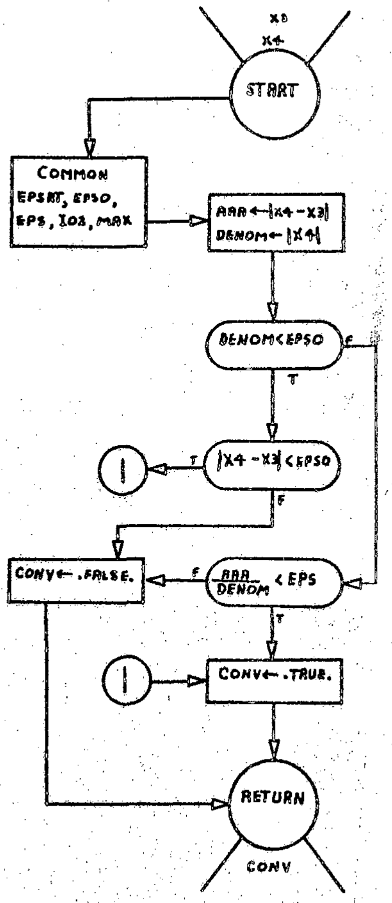


Figure 12.1. (Continued)

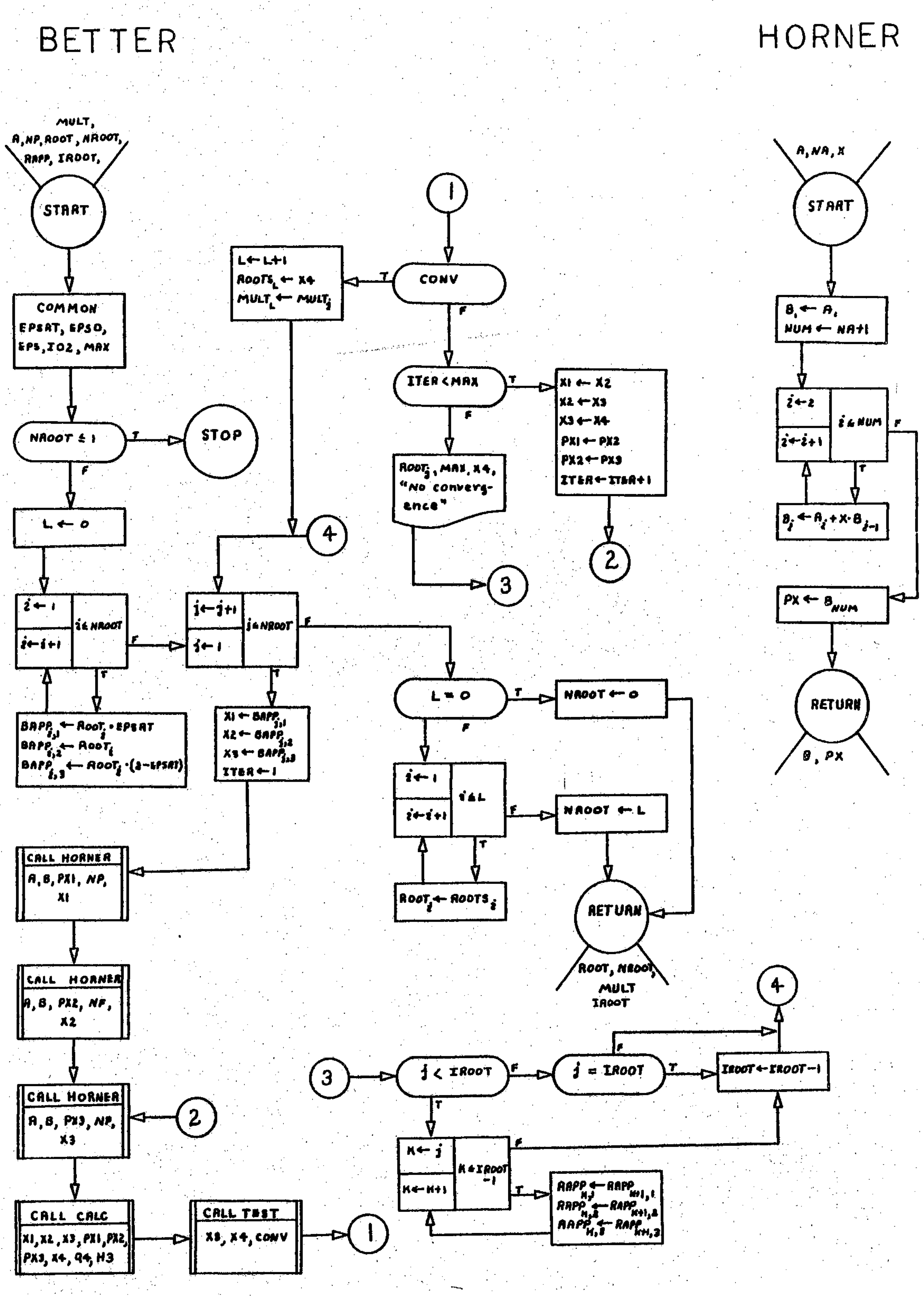
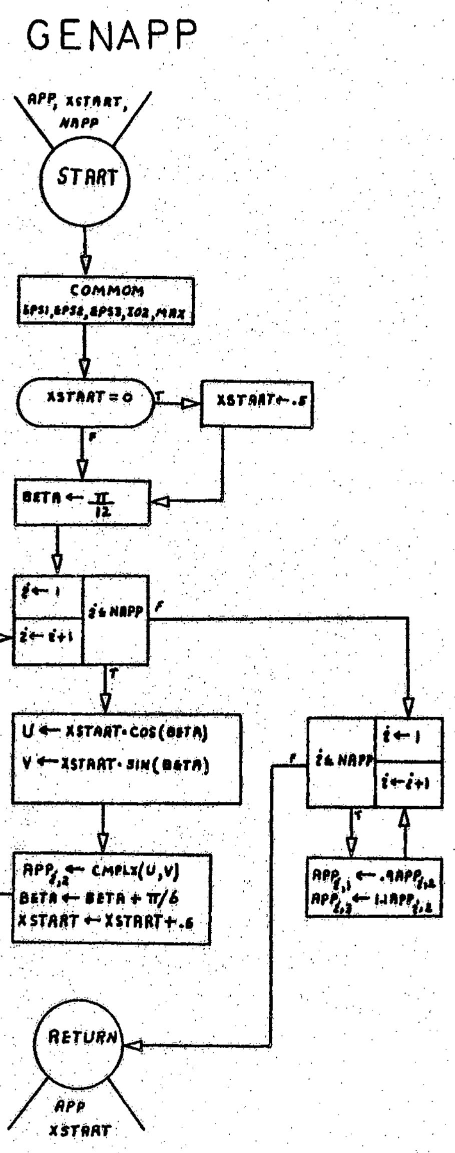


Figure 12.1. (Continued)



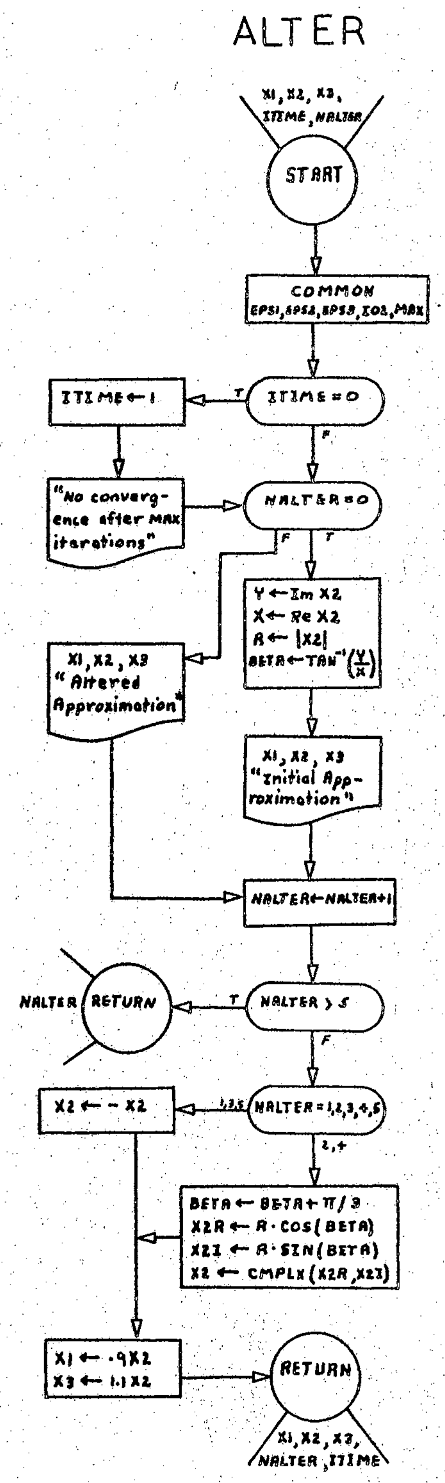
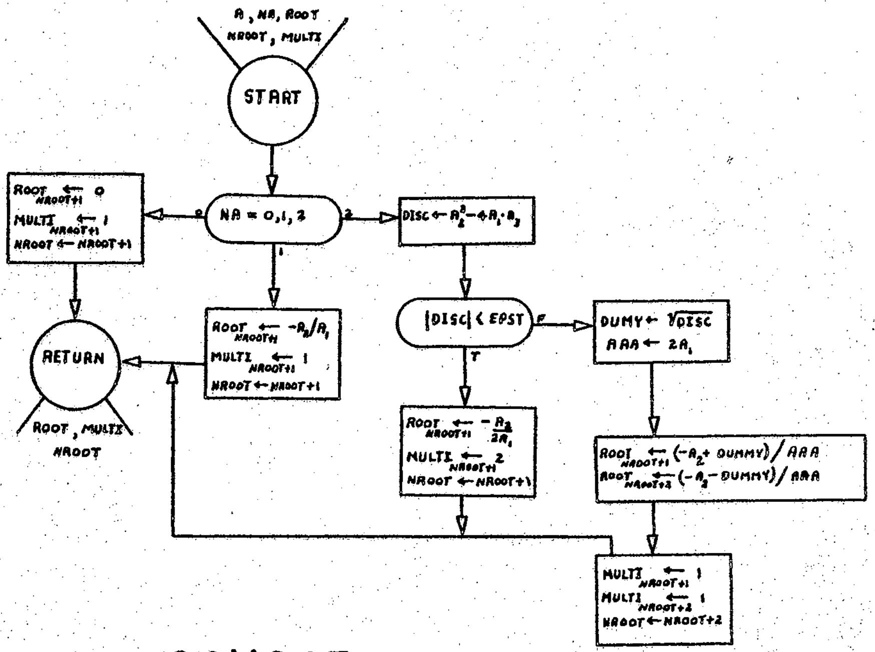


Figure 12.1. (Continued)

QUAD



COMSQT

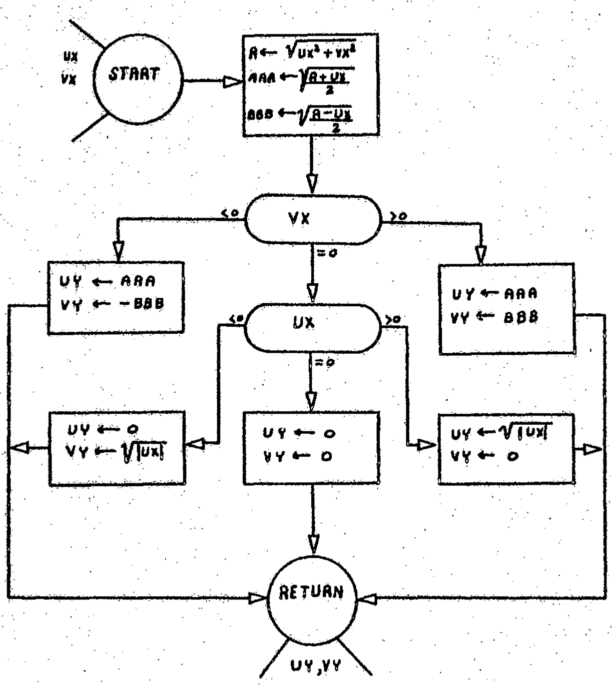


Figure 12.1. (Continued)

TABLE XIII-A

SINGLE PRECISION PROGRAM FOR MULLER'S METHOD

```
$JOB 10414
C     ***********************************************************************
C     *                                                                     *
C     * SINGLE PRECISION PROGRAM FOR MULLER'S METHOD                        *
C     *                                                                     *
C     *                                                                     *
C     * MULLER'S METHOD EXTRACTS THE ZEROS AND THEIR MULTIPLICITIES OF A    *
C     * POLYNOMIAL OF MAXIMUM DEGREE 25.  THROUGH THREE GIVEN POINTS THE    *
C     * POLYNOMIAL IS APPROXIMATED BY A QUADRATIC.  THE ZERO OF THE QUADRATIC *
C     * CLOSEST TO THE OLD APPROXIMATION IS TAKEN AS THE NEW APPROXIMATION. *
C     * IN THIS MANNER A SEQUENCE IS OBTAINED CONVERGING TO A ZERO.         *
C     *                                                                     *
C     ***********************************************************************
      COMPLEX PX3,PX2,ROOT,X1,APP,X2,WORK,X3,B,X4,A,PX1,RAPP,PX4,H3,Q4
      DIMENSION ROOT(25),MULT(25),APP(25,3),WORK(26),B(26),A(26),RAPP(25
     1,3)
      DATA PNAME,DNAME/2HP(,2HD(/
      LOGICAL CONV
      COMMON EPSRT,EPSO,EPS,IO2,MAX
      IO1=5
      IO2=6
      EPSRT=0.999
   10 NROOT=0
      IROOT=0
      IPATH=1
      NALTER=0
      ITIME=0
      NOMULT=0
      IAPP=1
      ITER=1
      READ(IO1,1000) NOPOLY,NP,NAPP,MAX,EPS,EPSO,EPSM,XSTART,XEND,KCHECK
      IF(KCHECK.EQ.1) STOP
      KKK=NP+1
      READ(IO1,1010) (A(I),I=1,KKK)
      WRITE(IO2,1020) NOPOLY,NP
      WRITE(IO2,1035) (PNAME,I,A(I),I=1,KKK)
      WRITE(IO2,2060)
      WRITE(IO2,2000) NAPP
      WRITE(IO2,2010) MAX
      WRITE(IO2,2020) EPS
      WRITE(IO2,2030) EPSM
      WRITE(IO2,2040) XSTART
      WRITE(IO2,2050) XEND
      IF(NP.GT.2) GO TO 15
      CALL QUAD(A,NP,ROOT,NROOT,MULT,EPSO)
      WRITE(IO2,1037)
      WRITE(IO2,1086) (I,ROOT(I),MULT(I),I=1,NROOT)
      GO TO 10
   15 IF(NAPP.NE.0) GO TO 20
      NAPP=NP
      CALL GENAPP(APP,NAPP,XSTART)
      GO TO 27
   20 READ(IO1,1030) (APP(I,2),I=1,NAPP)
      DO 25 I=1,NAPP
      APP(I,1)=0.9*APP(I,2)
   25 APP(I,3)=1.1*APP(I,2)
   27 KKK=NP+1
      DO 30 I=1,KKK
   30 WORK(I)=A(I)
```

TABLE XIII-A (Continued)

```
      NWORK=NP
   40 X1=APP(IAPP,1)
      X2=APP(IAPP,2)
      X3=APP(IAPP,3)
      CALL HORNER(NWORK,WORK,X1,B,PX1)
      CALL HORNER(NWORK,WORK,X2,B,PX2)
      CALL HORNER(NWORK,WORK,X3,B,PX3)
   50 CALL CALC(X1,X2,X3,PX1,PX2,PX3,X4,Q4,H3)
   55 CALL HORNER(NWORK,WORK,X4,B,PX4)
      AB1=CABS(PX3)
      IF(AB1.EQ.0.0) GO TO 60
      QQQ=CABS(PX4)/CABS(PX3)
      IF(QQQ.LE.10.) GO TO 60
      Q4=0.5*Q4
      X4=X3+H3*Q4
      GO TO 55
   60 CALL TEST(X3,X4,CONV)
      IF(CONV) GO TO 120
      IF(ITER.LT.MAX) GO TO 110
      CALL ALTER(APP(IAPP,1),APP(IAPP,2),APP(IAPP,3),NALTER,ITIME)
      IF(NALTER.GT.5) GO TO 75
      ITER=1
      GO TO 40
   75 IF(IAPP.LT.NAPP) GO TO 100
      IF(XEND.EQ.0.0) GO TO 70
      IF(XSTART.GT.XEND) GO TO 70
      NAPP=NP
      CALL GENAPP(APP,NAPP,XSTART)
      IAPP=0
      GO TO 100
   70 WRITE(IO2,1090)
      KKK=NWORK+1
      WRITE(IO2,1035) (DNAME,J,WORK(J),J=1,KKK)
   80 IF(NROOT.EQ.0) GO TO 90
      WRITE(IO2,1060)
      IF(IPATH.EQ.1) GO TO 82
   81 IPATH=2
      CALL BETTER(A,NP,ROOT,NROOT,RAPP,IROOT,MULT)
      WRITE(IO2,1200)
   82 IF(NROOT.EQ.0)GO TO 90
      IF(IROOT.EQ.0) GO TO 85
      WRITE(IO2,1080)
      DO 83 I=1,IROOT
   83 WRITE(IO2,1085) I,ROOT(I),MULT(I),RAPP(I,2)
      IF(IROOT.LT.NROOT) GO TO 85
      GO TO 87
   85 KKK=IROOT+1
      WRITE(IO2,1086) (I,ROOT(I),MULT(I),I=KKK,NROOT)
   87 IF(IPATH.EQ.1) GO TO 81
      GO TO 10
   90 WRITE(IO2,1070) NOPOLY
      GO TO 10
  100 IAPP=IAPP+1
      ITER=1
      NALTER=0
      GO TO 40
  120 NROOT=NROOT+1
      IROOT=NROOT
      MULT(NROOT)=1
      NOMULT=NOMULT+1
```

TABLE XIII-A (Continued)

```
      ROOT(NROOT)=X4
      RAPP(NROOT,1)=APP(IAPP,1)
      RAPP(NROOT,2)=APP(IAPP,2)
      RAPP(NROOT,3)=APP(IAPP,3)
  125 IF(NOMULT.LT.NP) GO TO 130
      GO TO 80
  130 CALL HORNER(NWORK,WORK,X4,B,PX4)
      NWORK=NWORK-1
      KKK=NWORK+1
      DO 140 I=1,KKK
  140 WORK(I)=B(I)
      CALL HORNER(NWORK,WORK,X4,B,PX4)
      CCC=CABS(PX4)
      IF(CCC.LT.EPSM) GO TO 150
      IF(NWORK.GT.2) GO TO 75
      IROOT=NROOT
      CALL QUAD(WORK,NWORK,ROOT,NROOT,MULT,EPSO)
      GO TO 80
  150 MULT(NROOT)=MULT(NROOT)+1
      NOMULT=NOMULT+1
      GO TO 125
  110 X1=X2
      X2=X3
      X3=X4
      PX1=PX2
      PX2=PX3
      PX3=PX4
      ITER=ITER+1
      GO TO 50
 1010 FORMAT(2(E30.0))
 1020 FORMAT(1H1,1X,52HMULLERS METHOD FOR FINDING THE ZEROS OF A POLYNOM
     1IAL/1H ,1X,18HPOLYNOMIAL NUMBER ,I2,11H OF DEGREE ,I2///1H ,1X,28H
     2THE COEFFICIENTS OF P(X) ARE//)
 1030 FORMAT(2(E30.0))
 1090 FORMAT(///,1X,65HCOEFFICIENTS OF DEFLATED POLYNOMIAL FOR WHICH NO
     1ZEROS WERE FOUND//)
 1080 FORMAT(///1X,13HROOTS OF P(X),37X,14HMULTIPLICITIES,11X,21HINITIAL
     1 APPROXIMATION//)
 1070 FORMAT(///1X,42HNO ZEROS WERE FOUND FOR POLYNOMIAL NUMBER ,I2)
 1037 FORMAT(///,1X,13HZEROS OF P(X),37X,14HMULTIPLICITIES//)
 1035 FORMAT(3X,A2,I2,4H) = ,E14.7,3H + ,E14.7,2H I)
 1085 FORMAT(2X,5HROOT(,I2,4H) = ,E14.7,3H + ,E14.7,2H I,10X,I2,10X,E14.
     17,3H + ,E14.7,2H I)
 1086 FORMAT(2X,5HROOT(,I2,4H) = ,E14.7,3H + ,E14.7,2H I,10X,I2,11X,23HS
     1OLVED BY DIRECT METHOD)
 1000 FORMAT(3(I2,1X),9X,I3,8X,3(E6.0,1X),13X,2(E7.0,1X),I1)
 2000 FORMAT(1X,41HNUMBER OF INITIAL APPROXIMATIONS GIVEN.  ,I2)
 2010 FORMAT(1X,29HMAXIMUM NUMBER OF ITERATIONS.,11X,I3)
 2020 FORMAT(1X,21HTEST FOR CONVERGENCE.,13X,E9.2)
 2030 FORMAT(1X,24HTEST FOR MULTIPLICITIES.,10X,E9.2)
 2040 FORMAT(1X,23HRADIUS TO START SEARCH.,11X,E9.2)
 2050 FORMAT(1X,21HRADIUS TO END SEARCH.,13X,E9.2)
 2060 FORMAT(//1X)
 1060 FORMAT(///35H BEFORE ATTEMPT TO IMPROVE ACCURACY)
 1200 FORMAT(///1X,37HAFTER THE ATTEMPT TO IMPROVE ACCURACY)
      END
```

TABLE XIII-A (Continued)

```
      SUBROUTINE BETTER(A,NP,ROOT,NROOT,RAPP,IROOT,MULT)
C    **************************************************************************
C    *                                                                        *
C    * SUBROUTINE BETTER ATTEMPTS TO IMPROVE THE ACCURACY OF THE ZEROS FOUND  *
C    * BY USING THEM AS INITIAL APPROXIMATIONS WITH MULLER'S METHOD APPLIED TO *
C    * THE FULL, UNDEFLATED POLYNOMIAL.                                       *
C    *                                                                        *
C    **************************************************************************
      COMPLEX ROOT,A,BAPP,X1,X2,X3,PX1,PX2,PX3,B,ROOTS,X4,RAPP,Q4,H3
      LOGICAL CONV
      DIMENSION ROOT(25),A(26),BAPP(25,3),B(26),ROOTS(25),RAPP(25,3),MUL
     1T(25)
      COMMON EPSRT,EPSO,EPS,IO2,MAX
      IF(NROOT.LE.1) RETURN
      L=0
      DO 10 I=1,NROOT
      BAPP(I,1)=ROOT(I)*EPSRT
      BAPP(I,2)=ROOT(I)
   10 BAPP(I,3)=ROOT(I)*(2.0-EPSRT)
      DO 100 J=1,NROOT
      X1=BAPP(J,1)
      X2=BAPP(J,2)
      X3=BAPP(J,3)
      ITER=1
      CALL HORNER(NP,A,X1,B,PX1)
      CALL HORNER(NP,A,X2,B,PX2)
   20 CALL HORNER(NP,A,X3,B,PX3)
      CALL CALC(X1,X2,X3,PX1,PX2,PX3,X4,Q4,H3)
   30 CALL TEST(X3,X4,CONV)
      IF(CONV) GO TO 50
      IF(ITER.LT.MAX) GO TO 40
      WRITE(IO2,1000) J,ROOT(J),MAX
      WRITE(IO2,1010) X4
      IF(J.LT.IROOT) GO TO 33
      IF(J.EQ.IROOT) GO TO 35
      GO TO 100
   33 KKK=IROOT-1
      DO 34 K=J,KKK
      RAPP(K,1)=RAPP(K+1,1)
      RAPP(K,2)=RAPP(K+1,2)
   34 RAPP(K,3)=RAPP(K+1,3)
   35 IROOT=IROOT-1
      GO TO 100
   40 X1=X2
      X2=X3
      X3=X4
      PX1=PX2
      PX2=PX3
      ITER=ITER+1
      GO TO 20
   50 L=L+1
      ROOTS(L)=X4
      MULT(L)=MULT(J)
  100 CONTINUE
      IF(L.EQ.0) GO TO 120
      DO 110 I=1,L
  110 ROOT(I)=ROOTS(I)
      NROOT=L
      RETURN
  120 NROOT=0
```

TABLE XIII-A (Continued)

```
      RETURN
 1000 FORMAT(///42H IN THE ATTEMPT TO IMPROVE ACCURACY, ROOT(,I2,4H) = ,
     1E14.7,3H + ,E14.7,2H I,24H DID NOT CONVERGE AFTER ,I3,11H ITERATIO
     2NS)
 1010 FORMAT(30H THE PRESENT APPROXIMATION IS ,E14.7,3H + ,E14.7,2H I//)
      END
```

```
      SUBROUTINE CALC(X1,X2,X3,PX1,PX2,PX3,X4,Q4,H3)
C   ****************************************************************************
C   *                                                                          *
C   * GIVEN THREE APPROXIMATIONS X(N-2), X(N-1), AND X(N), SUBROUTINE CALC     *
C   * APPROXIMATES THE POLYNOMIAL BY A QUADRATIC AND SOLVES FOR THE ZERO OF    *
C   * THE QUADRATIC CLOSEST TO X(N). THIS ZERO IS THE NEW APPROXIMATION        *
C   * X(N+1) TO THE ZERO OF THE POLYNOMIAL.                                    *
C   *                                                                          *
C   ****************************************************************************
      COMPLEX PX3,PX2,X1,X2,X3,PX1,H3,H2,Q3,D,B,C,DISC,DEN1,DEN2,Q4,X4
      COMPLEX TEST,CCC
      COMPLEX CMPLX
      COMMON EPSRT,EPSO,EPS,IO2,MAX
      H3=X3-X2
      H2=X2-X1
      Q3=H3/H2
      D=Q3*(PX3-((1.0+Q3)*PX2)+(Q3*PX1))
      B=(((2.0*Q3)+1.0)*PX3)-((1.0+Q3)*(1.0+Q3)*PX2)+(Q3*Q3*PX1)
      C=(1.0+Q3)*PX3
      DISC=(B*B)-(4.0*D*C)
      AAA=CABS(DISC)
      IF(AAA.EQ.0.0) GO TO 5
      GO TO 7
    5 THETA=0.0
      GO TO 9
    7 DISCI=AIMAG(DISC)
      DISCR=REAL(DISC)
      THETA=ATAN2(DISCI,DISCR)
    9 RAD=SQRT(AAA)
      ANGLE=THETA/2.0
      CCC=CMPLX(COS(ANGLE),SIN(ANGLE))
      TEST=RAD*CCC
      DEN1=B+TEST
      DEN2=B-TEST
      AAA=CABS(DEN1)
      BBB=CABS(DEN2)
      IF(AAA.LT.BBB) GO TO 10
      IF(AAA.EQ.0.0) GO TO 60
      Q4=(-2.0*C)/DEN1
      GO TO 50
   10 IF(BBB.EQ.0.0) GO TO 60
      Q4=(-2.0*C)/DEN2
      GO TO 50
   50 X4=X3+(H3*Q4)
      RETURN
   60 Q4=(1.0,0.0)
      GO TO 50
      END
```

TABLE XIII-A (Continued)

```
      SUBROUTINE GENAPP(APP,NAPP,XSTART)
C   *****************************************************************************
C   *                                                                           *
C   * SUBROUTINE GENAPP GENERATES N INITIAL APPROXIMATIONS, WHERE N IS THE      *
C   * DEGREE OF THE ORIGINAL POLYNOMIAL.                                        *
C   *                                                                           *
C   *****************************************************************************
      COMPLEX APP
      COMPLEX CMPLX
      DIMENSION APP(25,3)
      COMMON EPS1,EPS2,EPS3,IO2,MAX
      IF(XSTART.EQ.0.0) XSTART=0.5
      BETA=0.2617994
      DO 10 I=1,NAPP
      U=XSTART*COS(BETA)
      V=XSTART*SIN(BETA)
      APP(I,2)=CMPLX(U,V)
      BETA=BETA+0.5235988
   10 XSTART=XSTART+0.5
      DO 20 I=1,NAPP
      APP(I,1)=0.9*APP(I,2)
   20 APP(I,3)=1.1*APP(I,2)
      RETURN
      END
```

TABLE XIII-A (Continued)

```
      SUBROUTINE ALTER(X1,X2,X3,NALTER,ITIME)
C    **************************************************************************
C    *                                                                        *
C    * SUBROUTINE ALTER ALTERS THE INITIAL APPROXIMATIONS WHICH PRODUCE NO    *
C    * CONVERGENCE TO A ZERO. THIS IS DONE A MAXIMUM OF 5 TIMES FOR EACH ROOT. *
C    *                                                                        *
C    **************************************************************************
      COMPLEX X1,X2,X3
      COMPLEX CMPLX
      COMMON EPS1,EPS2,EPS3,IO2,MAX
      IF(ITIME.NE.0) GO TO 5
      ITIME=1
      WRITE(IO2,1010) MAX
    5 IF(NALTER.EQ.0) GO TO 10
      WRITE(IO2,1000) X1,X2,X3
      GO TO 20
   10 Y=AIMAG(X2)
      X=REAL(X2)
      R=CABS(X2)
      BETA=ATAN2(Y,X)
      WRITE(IO2,1020) X1,X2,X3
   20 NALTER=NALTER+1
      IF(NALTER.GT.5) RETURN
      GO TO (30,40,30,40,30),NALTER
   30 X2=-X2
      GO TO 50
   40 BETA=BETA+1.0471976
      X2R=R*COS(BETA)
      X2I=R*SIN(BETA)
      X2=CMPLX(X2R,X2I)
   50 X1=0.9*X2
      X3=1.1*X2
      RETURN
 1010 FORMAT(///1X,54HNO CONVERGENCE FOR THE FOLLOWING APPROXIMATIONS AF
     1TER ,I3,12H ITERATIONS.//)
 1000 FORMAT(1X,5HX1 = ,E14.7,3H + ,E14.7,2H I,10X,22HALTERED APPROXIMAT
     1IONS/1X,5HX2 = ,E14.7,3H + ,E14.7,2H I/1X,5HX3 = ,E14.7,3H + ,E14.
     27,2H I/)
 1020 FORMAT(1H0,5HX1 = ,E14.7,3H + ,E14.7,2H I,10X,22HINITIAL APPROXIMA
     1TIONS/1X,5HX2 = ,E14.7,3H + ,E14.7,2H I/1X,5HX3 = ,E14.7,3H + ,E14
     2.7,2H I/)
      END
```

TABLE XIII-A (Continued)

```
      SUBROUTINE TEST(X3,X4,CONV)
C    ***********************************************************************
C    *                                                                     *
C    * SUBROUTINE TEST CHECKS FOR CONVERGENCE OF THE SEQUENCE OF APPROX-   *
C    * IMATIONS BY TESTING THE EXPRESSION                                  *
C    * ABSOLUTE VALUE OF (X(N+1)-X(N))/ABSOLUTE VALUE OF X(N+1).           *
C    * WHEN IT IS AS SMALL AS DESIRED, CONVERGENCE IS OBTAINED.            *
C    *                                                                     *
C    ***********************************************************************
      COMPLEX X3,X4
      LOGICAL CONV
      COMMON EPSRT,EPSO,EPS,IO2,MAX
      AAA=CABS(X4-X3)
      DENOM=CABS(X4)
      IF(DENOM.LT.EPSO) GO TO 20
      IF(AAA/DENOM.LT.EPS) GO TO 10
    5 CONV=.FALSE.
      GO TO 100
   10 CONV=.TRUE.
      GO TO 100
   20 IF(AAA.LT.EPSO) GO TO 10
      GO TO 5
  100 RETURN
      END
```

```
      SUBROUTINE HORNER(NA,A,X,B,PX)
C    ***********************************************************************
C    *                                                                     *
C    * HORNER'S METHOD COMPUTES THE VALUE OF THE POLYNOMIAL P(X) AT A POINT D.  *
C    * SYNTHETIC DIVISION IS USED TO DEFLATE THE POLYNOMIAL BY DIVIDING OUT THE *
C    * FACTOR (X-D).                                                       *
C    *                                                                     *
C    ***********************************************************************
      COMPLEX X,PX,B,A
      DIMENSION A(26),B(26)
      B(1)=A(1)
      NUM=NA+1
      DO 10 I=2,NUM
   10 B(I)=A(I)+B(I-1)*X
      PX=B(NUM)
      RETURN
      END
```

TABLE XIII-A (Continued)

```
      SUBROUTINE QUAD(A,NA,ROOT,NROOT,MULTI,EPST)
C     ***********************************************************************
C     *                                                                     *
C     * SUBROUTINE QUAD SOLVES DIRECTLY FOR THE ZEROS AND THEIR MULTIPLICITIES *
C     * OF EITHER A QUADRATIC POLYNOMIAL OR A LINEAR FACTOR.  SOLUTION OF THE  *
C     * QUADRATIC IS DONE USING THE QUADRATIC FORMULA.                       *
C     *                                                                     *
C     ***********************************************************************
      COMPLEX A,DISC,ROOT,DUMMY,AAA
      COMPLEX CSQRT
      DIMENSION A(26),ROOT(25),MULTI(25)
      IF(NA.EQ.2) GO TO 7
      IF(NA.EQ.1) GO TO 5
      ROOT(NROOT+1)=0.0
      MULTI(NROOT+1)=1
      NROOT=NROOT+1
      GO TO 50
    5 ROOT(NROOT+1)=-A(2)/A(1)
      MULTI(NROOT+1)=1
      NROOT=NROOT+1
      GO TO 50
    7 DISC=A(2)*A(2)-(4.0*A(1)*A(3))
      BBB=CABS(DISC)
      IF(BBB.LT.EPST) GO TO 10
      DUMMY=CSQRT(DISC)
      AAA=2.0*A(1)
      ROOT(NROOT+1)=(-A(2)+DUMMY)/AAA
      ROOT(NROOT+2)=(-A(2)-DUMMY)/AAA
      MULTI(NROOT+1)=1
      MULTI(NROOT+2)=1
      NROOT=NROOT+2
      GO TO 50
   10 ROOT(NROOT+1)=(-A(2))/(2.0*A(1))
      MULTI(NROOT+1)=2
      NROOT=NROOT+1
   50 RETURN
      END

$ENTRY
```

TABLE XIII-B

DOUBLE PRECISION PROGRAM FOR MULLER'S METHOD

```
$JOB 10414
C   ****************************************************************************
C   *                                                                          *
C   * DOUBLE PRECISION PROGRAM FOR MULLER'S METHOD                             *
C   *                                                                          *
C   *                                                                          *
C   * MULLER'S METHOD EXTRACTS THE ZEROS AND THEIR MULTIPLICITIES OF A         *
C   * POLYNOMIAL OF MAXIMUM DEGREE 25.  THROUGH THREE GIVEN POINTS THE         *
C   * POLYNOMIAL IS APPROXIMATED BY A QUADRATIC.  THE ZERO OF THE QUADRATIC    *
C   * CLOSEST TO THE OLD APPROXIMATION IS TAKEN AS THE NEW APPROXIMATION.      *
C   * IN THIS MANNER A SEQUENCE IS OBTAINED CONVERGING TO A ZERO.              *
C   *                                                                          *
C   ****************************************************************************
      DOUBLE PRECISION UPX3,VPX3,UPX2,VPX2,UROOT,VROOT,UX1,VX1,UAPP,VAPP
     1,UX2,VX2,UWORK,VWORK,UX3,VX3,UB,VB,UX4,VX4,UA,VA,UPX1,VPX1,URAPP,V
     2RAPP,UPX4,VPX4,EPSRT,EPSO,EPS,CCC,EPSM,UH3,VH3,UQ4,VQ4,ABPX4,ABPX3
     3,QQQ,XSTART,XEND
      DOUBLE PRECISION DSQRT
      DIMENSION UROOT(25),VROOT(25),MULT(25),UAPP(25,3),VAPP(25,3),UWORK
     1(26),VWORK(26),UB(26),VB(26),UA(26),VA(26),URAPP(25,3),VRAPP(25,3)
      DATA PNAME,DNAME/2HP(,2HD(/
      LOGICAL CONV
      COMMON EPSRT,EPSO,EPS,IO2,MAX
      IO1=5
      IO2=6
      EPSRT=0.999
   10 NROOT=0
      IROOT=0
      IPATH=1
      NOMULT=0
      NALTER=0
      ITIME=0
      IAPP=1
      ITER=1
      READ(IO1,1000) NOPOLY,NP,NAPP,MAX,EPS,EPSO,EPSM,XSTART,XEND,KCHECK
      IF(KCHECK.EQ.1) STOP
      KKK=NP+1
      READ(IO1,1010) (UA(I),VA(I),I=1,KKK)
      WRITE(IO2,1020) NOPOLY,NP
      WRITE(IO2,1035) (PNAME,I,UA(I),VA(I),I=1,KKK)
      WRITE(IO2,2060)
      WRITE(IO2,2000) NAPP
      WRITE(IO2,2010) MAX
      WRITE(IO2,2020) EPS
      WRITE(IO2,2030) EPSM
      WRITE(IO2,2040) XSTART
      WRITE(IO2,2050) XEND
      IF(NP.GT.2) GO TO 15
      CALL QUAD(UA,VA,NP,UROOT,VROOT,NROOT,MULT,EPSO)
      WRITE(IO2,1037)
      WRITE(IO2,1086) (I,UROOT(I),VROOT(I),MULT(I),I=1,NROOT)
      GO TO 10
   15 IF(NAPP.NE.0) GO TO 20
      NAPP=NP
      CALL GENAPP(UAPP,VAPP,NAPP,XSTART)
      GO TO 27
   20 READ(IO1,1030) (UAPP(I,2),VAPP(I,2),I=1,NAPP)
      DO 25 I=1,NAPP
      UAPP(I,1)=0.9*UAPP(I,2)
```

TABLE XIII-B (Continued)

```
      VAPP(I,1)=0.9*VAPP(I,2)
      UAPP(I,3)=1.1*UAPP(I,2)
   25 VAPP(I,3)=1.1*VAPP(I,2)
   27 KKK=NP+1
      DO 30 I=1,KKK
      UWORK(I)=UA(I)
   30 VWORK(I)=VA(I)
      NWORK=NP
   40 UX1=UAPP(IAPP,1)
      VX1=VAPP(IAPP,1)
      UX2=UAPP(IAPP,2)
      VX2=VAPP(IAPP,2)
      UX3=UAPP(IAPP,3)
      VX3=VAPP(IAPP,3)
      CALL HORNER(NWORK,UWORK,VWORK,UX1,VX1,UB,VB,UPX1,VPX1)
      CALL HORNER(NWORK,UWORK,VWORK,UX2,VX2,UB,VB,UPX2,VPX2)
      CALL HORNER(NWORK,UWORK,VWORK,UX3,VX3,UB,VB,UPX3,VPX3)
   50 CALL CALC(UX1,VX1,UX2,VX2,UX3,VX3,UPX1,VPX1,UPX2,VPX2,UPX3,VPX3,UX
     14,VX4,UQ4,VQ4,UH3,VH3)
   60 CALL HORNER(NWORK,UWORK,VWORK,UX4,VX4,UB,VB,UPX4,VPX4)
      ABPX4=DSQRT(UPX4*UPX4+VPX4*VPX4)
      ABPX3=DSQRT(UPX3*UPX3+VPX3*VPX3)
      IF(ABPX3.EQ.0.0) GO TO 70
      QQQ=ABPX4/ABPX3
      IF(QQQ.LE.10.) GO TO 70
      UQ4=0.5*UQ4
      VQ4=0.5*VQ4
      UX4=UX3+(UH3*UQ4-VH3*VQ4)
      VX4=VX3+(VH3*UQ4+UH3*VQ4)
      GO TO 60
   70 CALL TEST(UX3,VX3,UX4,VX4,CONV)
      IF(CONV) GO TO 120
      IF(ITER.LT.MAX) GO TO 110
      CALL ALTER(UAPP(IAPP,1),VAPP(IAPP,1),UAPP(IAPP,2),VAPP(IAPP,2),UAP
     1P(IAPP,3),VAPP(IAPP,3),NALTER,ITIME)
      IF(NALTER.GT.5) GO TO 75
      ITER=1
      GO TO 40
   75 IF(IAPP.LT.NAPP) GO TO 100
      IF(XEND.EQ.0.0) GO TO 77
      IF(XSTART.GT.XEND) GO TO 77
      NAPP=NP
      CALL GENAPP(UAPP,VAPP,NAPP,XSTART)
      IAPP=0
      GO TO 100
   77 WRITE(IO2,1090)
      KKK=NWORK+1
      WRITE(IO2,1035) (DNAME,J,UWORK(J),VWORK(J),J=1,KKK)
   80 IF(NROOT.EQ.0) GO TO 90
      WRITE(IO2,1060)
      IF(IPATH.EQ.1) GO TO 82
   81 IPATH=2
      CALL BETTER(UA,VA,NP,UROOT,VROOT,NROOT,URAPP,VRAPP,IROOT,MULT)
      WRITE(IO2,1200)
   82 IF(NROOT.EQ.0)GO TO 90
      IF(IROOT.EQ.0) GO TO 85
      WRITE(IO2,1080)
      DO 55 I=1,IROOT
   55 WRITE(IO2,1085) I,UROOT(I),VROOT(I),MULT(I),URAPP(I,2),VRAPP(I,2)
      IF(IROOT.LT.NROOT) GO TO 85
```

TABLE XIII-B (Continued)

```
      GO TO 87
   85 KKK=IROOT+1
      WRITE(IO2,1086) (I,UROOT(I),VROOT(I),MULT(I),I=KKK,NROOT)
   87 IF(IPATH.EQ.1) GO TO 81
      GO TO 10
   90 WRITE(IO2,1070) NOPOLY
      GO TO 10
  100 IAPP=IAPP+1
      ITER=1
      NALTER=0
      GO TO 40
  120 NROOT=NROOT+1
      IROOT=NROOT
      MULT(NROOT)=1
      NOMULT=NOMULT+1
      UROOT(NROOT)=UX4
      VROOT(NROOT)=VX4
      URAPP(NROOT,1)=UAPP(IAPP,1)
      VRAPP(NROOT,1)=VAPP(IAPP,1)
      URAPP(NROOT,2)=UAPP(IAPP,2)
      VRAPP(NROOT,2)=VAPP(IAPP,2)
      URAPP(NROOT,3)=UAPP(IAPP,3)
      VRAPP(NROOT,3)=VAPP(IAPP,3)
  125 IF(NOMULT.LT.NP) GO TO 130
      GO TO 80
  130 CALL HORNER(NWORK,UWORK,VWORK,UX4,VX4,UB,VB,UPX4,VPX4)
      NWORK=NWORK-1
      KKK=NWORK+1
      DO 140 I=1,KKK
      UWORK(I)=UB(I)
  140 VWORK(I)=VB(I)
      CALL HORNER(NWORK,UWORK,VWORK,UX4,VX4,UB,VB,UPX4,VPX4)
      CCC=DSQRT(UPX4*UPX4+VPX4*VPX4)
      IF(CCC.LT.EPSM) GO TO 150
      IF(NWORK.GT.2) GO TO 75
      IROOT=NROOT
      CALL QUAD(UWORK,VWORK,NWORK,UROOT,VROOT,NROOT,MULT,EPSO)
      GO TO 80
  150 MULT(NROOT)=MULT(NROOT)+1
      NOMULT=NOMULT+1
      GO TO 125
  110 UX1=UX2
      VX1=VX2
      UX2=UX3
      VX2=VX3
      UX3=UX4
      VX3=VX4
      UPX1=UPX2
      VPX1=VPX2
      UPX2=UPX3
      VPX2=VPX3
      UPX3=UPX4
      VPX3=VPX4
      ITER=ITER+1
      GO TO 50
 1010 FORMAT(2D30.0)
 1020 FORMAT(1H1,1X,52HMULLERS METHOD FOR FINDING THE ZEROS OF A POLYNOM
     1IAL/1H ,1X,18HPOLYNOMIAL NUMBER ,I2,11H OF DEGREE ,I2///1H ,1X,28H
     2THE COEFFICIENTS OF P(X) ARE//)
 1030 FORMAT(2D30.0)
```

TABLE XIII-B (Continued)

```
 1090 FORMAT(///,1X,65HCOEFFICIENTS OF DEFLATED POLYNOMIAL FOR WHICH NO
     1ZEROS WERE FOUND//)
 1080 FORMAT(///1X,13HROOTS OF P(X),52X,14HMULTIPLICITIES,17X,21HINITIAL
     1 APPROXIMATION//)
 1070 FORMAT(//,43H NO ZEROS WERE FOUND FOR POLYNOMIAL NUMBER ,I2)
 1086 FORMAT(2X,5HROOT(,I2,4H) = ,D23.16,3H + ,D23.16,2H I,8X,I2,9X,23HS
     1OLVED BY DIRECT METHOD)
 1037 FORMAT(///,1X,13HZEROS OF P(X),51X,14HMULTIPLICITIES//)
 1035 FORMAT(3X,A2,I2,4H) = ,D23.16,3H + ,D23.16,2H I)
 1085 FORMAT(2X,5HROOT(,I2,4H) = ,D23.16,3H + ,D23.16,2H I,8X,I2,8X,D23.
     116,3H + ,D23.16,2H I)
 1000 FORMAT(3(I2,1X),9X,I3,8X,3(D6.0,1X),13X,2(D7.0,1X),I1)
 1060 FORMAT(///35H BEFORE ATTEMPT TO IMPROVE ACCURACY)
 1200 FORMAT(///1X,37HAFTER THE ATTEMPT TO IMPROVE ACCURACY)
 2000 FORMAT(1X,41HNUMBER OF INITIAL APPROXIMATIONS GIVEN.  ,I2)
 2010 FORMAT(1X,29HMAXIMUM NUMBER OF ITERATIONS.,11X,I3)
 2020 FORMAT(1X,21HTEST FOR CONVERGENCE.,13X,D9.2)
 2030 FORMAT(1X,24HTEST FOR MULTIPLICITIES.,10X,D9.2)
 2040 FORMAT(1X,23HRADIUS TO START SEARCH.,11X,D9.2)
 2050 FORMAT(1X,21HRADIUS TO END SEARCH.,13X,D9.2)
 2060 FORMAT(//1X)
      END
```

TABLE XIII-B (Continued)

```
      SUBROUTINE ALTER(X1R,X1I,X2R,X2I,X3R,X3I,NALTER,ITIME)
C   ***************************************************************************
C   *                                                                         *
C   * SUBROUTINE ALTER ALTERS THE INITIAL APPROXIMATIONS WHICH PRODUCE NO     *
C   * CONVERGENCE TO A ZERO. THIS IS DONE A MAXIMUM OF 5 TIMES FOR EACH ROOT. *
C   *                                                                         *
C   ***************************************************************************
      DOUBLE PRECISION X1R,X1I,X2R,X2I,X3R,X3I,EPS1,EPS2,EPS3,R,BETA
      DOUBLE PRECISION DCOS,DSIN
      DOUBLE PRECISION DSQRT
      DOUBLE PRECISION DATAN2
      COMMON EPS1,EPS2,EPS3,IO2,MAX
      IF(ITIME.NE.0) GO TO 5
      ITIME=1
      WRITE(IO2,1010) MAX
    5 IF(NALTER.EQ.0) GO TO 10
      WRITE(IO2,1000) X1R,X1I,X2R,X2I,X3R,X3I
      GO TO 20
   10 R=DSQRT(X2R*X2R+X2I*X2I)
      BETA=DATAN2(X2I,X2R)
      WRITE(IO2,1020) X1R,X1I,X2R,X2I,X3R,X3I
   20 NALTER=NALTER+1
      IF(NALTER.GT.5) RETURN
      GO TO (30,40,30,40,30),NALTER
   30 X2R=-X2R
      X2I=-X2I
      GO TO 50
   40 BETA=BETA+1.0471976
      X2R=R*DCOS(BETA)
      X2I=R*DSIN(BETA)
   50 X1R=0.9*X2R
      X1I=0.9*X2I
      X3R=1.1*X2R
      X3I=1.1*X2I
      RETURN
 1000 FORMAT(1X,5HX1 = ,D23.16,3H + ,D23.16,2H I,10X,22HALTERED APPROXIM
     1ATIONS/1X,5HX2 = ,D23.16,3H + ,D23.16,2H I/1X,5HX3 = ,D23.16,3H +
     2,D23.16,2H I/)
 1020 FORMAT(1H0,5HX1 = ,D23.16,3H + ,D23.16,2H I,10X,22HINITIAL APPROXI
     1MATIONS/1X,5HX2 = ,D23.16,3H + ,D23.16,2H I/1X,5HX3 = ,D23.16,3H +
     2 ,D23.16,2H I/)
 1010 FORMAT(///1X,54HNO CONVERGENCE FOR THE FOLLOWING APPROXIMATIONS AF
     1TER ,I3,12H ITERATIONS.//)
      END
```

TABLE XIII-B (Continued)

```
      SUBROUTINE GENAPP(APPR,APPI,NAPP,XSTART)
C   ****************************************************************************
C   *                                                                          *
C   * SUBROUTINE GENAPP GENERATES N INITIAL APPROXIMATIONS, WHERE N IS THE     *
C   * DEGREE OF THE ORIGINAL POLYNOMIAL.                                       *
C   *                                                                          *
C   ****************************************************************************
      DOUBLE PRECISION APPR,APPI,XSTART,EPS1,EPS2,EPS3,BETA
      DOUBLE PRECISION DCOS,DSIN
      DIMENSION APPR(25,3),APPI(25,3)
      COMMON EPS1,EPS2,EPS3,IO2,MAX
      IF(XSTART.EQ.0.0) XSTART=0.5
      BETA=0.2617994
      DO 10 I=1,NAPP
      APPR(I,2)=XSTART*DCOS(BETA)
      APPI(I,2)=XSTART*DSIN(BETA)
      BETA=BETA+0.5235988
   10 XSTART=XSTART+0.5
      DO 20 I=1,NAPP
      APPR(I,1)=0.9*APPR(I,2)
      APPI(I,1)=0.9*APPI(I,2)
      APPR(I,3)=1.1*APPR(I,2)
   20 APPI(I,3)=1.1*APPI(I,2)
      RETURN
      END
```

TABLE XIII-B (Continued)

```
      SUBROUTINE BETTER(UA,VA,NP,UROOT,VROOT,NROOT,URAPP,VRAPP,IROOT,MUL
     1T)
C    ************************************************************************
C    *                                                                      *
C    * SUBROUTINE BETTER ATTEMPTS TO IMPROVE THE ACCURACY OF THE ZEROS FOUND   *
C    * BY USING THEM AS INITIAL APPROXIMATIONS WITH MULLER'S METHOD APPLIED TO *
C    * THE FULL, UNDEFLATED POLYNOMIAL.                                      *
C    *                                                                      *
C    ************************************************************************
      DOUBLE PRECISION UROOT,VROOT,UA,VA,UBAPP,VBAPP,UX1,VX1,UX2,VX2,UX3
     1,VX3,UPX1,VPX1,UPX2,VPX2,UPX3,VPX3,UB,VB,UROOTS,VROOTS,EPSRT,UX4,V
     2X4,URAPP,VRAPP,EPSO,EPS,UQ4,VQ4,UH3,VH3
      LOGICAL CONV
      DIMENSION UROOT(25),VROOT(25),UA(26),VA(26),UBAPP(25,3),VBAPP(25,3
     1),UB(26),VB(26),UROOTS(25),VROOTS(25),URAPP(25,3),VRAPP(25,3),MULT
     3(25)
      COMMON EPSRT,EPSO,EPS,IO2,MAX
      IF(NROOT.LE.1) RETURN
      L=0
      DO 10 I=1,NROOT
      UBAPP(I,1)=UROOT(I)*EPSRT
      VBAPP(I,1)=VROOT(I)*EPSRT
      UBAPP(I,2)=UROOT(I)
      VBAPP(I,2)=VROOT(I)
      UBAPP(I,3)=UROOT(I)*(2.0-EPSRT)
   10 VBAPP(I,3)=VROOT(I)*(2.0-EPSRT)
      DO 100 J=1,NROOT
      UX1=UBAPP(J,1)
      VX1=VBAPP(J,1)
      UX2=UBAPP(J,2)
      VX2=VBAPP(J,2)
      UX3=UBAPP(J,3)
      VX3=VBAPP(J,3)
      ITER=1
      CALL HORNER(NP,UA,VA,UX1,VX1,UB,VB,UPX1,VPX1)
      CALL HORNER(NP,UA,VA,UX2,VX2,UB,VB,UPX2,VPX2)
   20 CALL HORNER(NP,UA,VA,UX3,VX3,UB,VB,UPX3,VPX3)
      CALL CALC(UX1,VX1,UX2,VX2,UX3,VX3,UPX1,VPX1,UPX2,VPX2,UPX3,VPX3,UX
     14,VX4,UQ4,VQ4,UH3,VH3)
   30 CALL TEST(UX3,VX3,UX4,VX4,CONV)
      IF(CONV) GO TO 50
      IF(ITER.LT.MAX) GO TO 40
      WRITE(IO2,1000) J,UROOT(J),VROOT(J),MAX
      WRITE(IO2,1010) UX4,VX4
      IF(J.LT.IROOT) GO TO 33
      IF(J.EQ.IROOT) GO TO 35
      GO TO 100
   33 KKK=IROOT-1
      DO 34 K=J,KKK
      URAPP(K,1)=URAPP(K+1,1)
      VRAPP(K,1)=VRAPP(K+1,1)
      URAPP(K,2)=URAPP(K+1,2)
      VRAPP(K,2)=VRAPP(K+1,2)
      URAPP(K,3)=URAPP(K+1,3)
   34 VRAPP(K,3)=VRAPP(K+1,3)
   35 IROOT=IROOT-1
      GO TO 100
   40 UX1=UX2
      VX1=VX2
      UX2=UX3
```

TABLE XIII-B (Continued)

```
      VX2=VX3
      UX3=UX4
      VX3=VX4
      UPX1=UPX2
      VPX1=VPX2
      UPX2=UPX3
      VPX2=VPX3
      ITER=ITER+1
      GO TO 20
   50 L=L+1
      UROOTS(L)=UX4
      VROOTS(L)=VX4
      MULT(L)=MULT(J)
  100 CONTINUE
      IF(L.EQ.0) GO TO 120
      DO 110 I=1,L
      UROOT(I)=UROOTS(I)
  110 VROOT(I)=VROOTS(I)
      NROOT=L
      RETURN
  120 NROOT=0
      RETURN
 1000 FORMAT(///42H IN THE ATTEMPT TO IMPROVE ACCURACY, ROOT(,I2,4H) = ,
     1D23.16,3H + ,D23.16,2H I/24H DID NOT CONVERGE AFTER ,I3,11H ITERAT
     2IONS)
 1010 FORMAT(30H THE PRESENT APPROXIMATION IS ,D23.16,3H + ,D23.16,2H I/
     1/)
      END
```

TABLE XIII-B (Continued)

```
      SUBROUTINE CALC(UX1,VX1,UX2,VX2,UX3,VX3,UPX1,VPX1,UPX2,VPX2,UPX3,V
     1PX3,UX4,VX4,UQ4,VQ4,UH3,VH3)
C    ********************************************************************
C    *                                                                  *
C    * GIVEN THREE APPROXIMATIONS X(N-2), X(N-1), AND X(N), SUBROUTINE CALC  *
C    * APPROXIMATES THE POLYNOMIAL BY A QUADRATIC AND SOLVES FOR THE ZERO OF  *
C    * THE QUADRATIC CLOSEST TO X(N). THIS ZERO IS THE NEW APPROXIMATION    *
C    * X(N+1) TO THE ZERO OF THE POLYNOMIAL.                             *
C    *                                                                  *
C    ********************************************************************
      DOUBLE PRECISION ARG1,ARG2
      DOUBLE PRECISION UPX3,VPX3,UPX2,VPX2,UX1,VX1,UX2,VX2,UX3,VX3,UPX1,
     1VPX1,UH3,VH3,UH2,VH2,UQ3,VQ3,UD,VD,UB,VB,UC,VC,UDISC,VDISC,UCCC,VC
     2CC,UDEN1,VDEN1,UDEN2,VDEN2,UQ4,VQ4,UX4,VX4,EPSRT,EPSO,EPS,UDDD,VDD
     3D,AAA,BBB,RAD,UAAA,VAAA,UBBB,VBBB
      DOUBLE PRECISION THETA,ANGLE,UTEST,VTEST
      DOUBLE PRECISION DATAN2,DCOS,DSIN,DSQRT
      COMMON EPSRT,EPSO,EPS,IO2,MAX
      UH3=UX3-UX2
      VH3=VX3-VX2
      UH2=UX2-UX1
      VH2=VX2-VX1
      BBB=UH2*UH2+VH2*VH2
      UQ3=(UH3*UH2+VH3*VH2)/BBB
      VQ3=(VH3*UH2-UH3*VH2)/BBB
      UDDD=1.0+UQ3
      VDDD=VQ3
      UD=(UPX3-(UDDD*UPX2-VDDD*VPX2))+(UQ3*UPX1-VQ3*VPX1)
      VD=(VPX3-(VDDD*UPX2+UDDD*VPX2))+(VQ3*UPX1+UQ3*VPX1)
      UAAA=2.0*UQ3
      VAAA=2.0*VQ3
      UAAA=UAAA+1.0
      UBBB=UDDD*UDDD-VDDD*VDDD
      VBBB=VDDD*UDDD+UDDD*VDDD
      UCCC=UQ3*UQ3-VQ3*VQ3
      VCCC=VQ3*UQ3+UQ3*VQ3
      UB=((UAAA*UPX3-VAAA*VPX3)-(UBBB*UPX2-VBBB*VPX2))+(UCCC*UPX1-VCCC*V
     1PX1)
      VB=((VAAA*UPX3+UAAA*VPX3)-(VBBB*UPX2+UBBB*VPX2))+(VCCC*UPX1+UCCC*V
     1PX1)
      UC=UDDD*UPX3-VDDD*VPX3
      VC=VDDD*UPX3+UDDD*VPX3
      UDISC=(UB*UB-VB*VB)-(4.0*(UD*UC-VD*VC))
      VDISC=(2.0*(VB*UB))-(4.0*(VD*UC+UD*VC))
      AAA=DSQRT(UDISC*UDISC+VDISC*VDISC)
      IF(AAA.EQ.0.0) GO TO 5
      GO TO 7
    5 THETA=0.0
      GO TO 9
    7 THETA=DATAN2(VDISC,UDISC)
    9 RAD=DSQRT(AAA)
      ANGLE=THETA/2.0
      UTEST=RAD*DCOS(ANGLE)
      VTEST=RAD*DSIN(ANGLE)
      UDEN1=UB+UTEST
      VDEN1=VB+VTEST
      UDEN2=UB-UTEST
      VDEN2=VB-VTEST
      ARG1=UDEN1*UDEN1+VDEN1*VDEN1
      ARG2=UDEN2*UDEN2+VDEN2*VDEN2
```

TABLE XIII-B (Continued)

```
      AAA=DSQRT(ARG1)
      BBB=DSQRT(ARG2)
      IF(AAA.LT.BBB) GO TO 10
      IF(AAA.EQ.0.0) GO TO 60
      UAAA=-2.0*UC
      VAAA=-2.0*VC
      UQ4=(UAAA*UDEN1+VAAA*VDEN1)/ARG1
      VQ4=(VAAA*UDEN1-UAAA*VDEN1)/ARG1
      GO TO 50
   10 IF(BBB.EQ.0.0) GO TO 60
      UAAA=-2.0*UC
      VAAA=-2.0*VC
      UQ4=(UAAA*UDEN2+VAAA*VDEN2)/ARG2
      VQ4=(VAAA*UDEN2-UAAA*VDEN2)/ARG2
      GO TO 50
   50 UX4=UX3+(UH3*UQ4-VH3*VQ4)
      VX4=VX3+(VH3*UQ4+UH3*VQ4)
      RETURN
   60 UQ4=1.0
      VQ4=0.0
      GO TO 50
      END
```

```
      SUBROUTINE TEST(UX3,VX3,UX4,VX4,CONV)
C   ****************************************************************************
C   *                                                                          *
C   * SUBROUTINE TEST CHECKS FOR CONVERGENCE OF THE SEQUENCE OF APPROX-        *
C   * IMATIONS BY TESTING THE EXPRESSION                                       *
C   * ABSOLUTE VALUE OF (X(N+1)-X(N))/ABSOLUTE VALUE OF X(N+1).                *
C   * WHEN IT IS AS SMALL AS DESIRED, CONVERGENCE IS OBTAINED.                 *
C   *                                                                          *
C   ****************************************************************************
      DOUBLE PRECISION DSQRT
      DOUBLE PRECISION UX3,VX3,UX4,VX4,EPSRT,EPSO,EPS,AAA,UDUMMY,VDUMMY,
     1DENOM
      LOGICAL CONV
      COMMON EPSRT,EPSO,EPS,IO2,MAX
      UDUMMY=UX4-UX3
      VDUMMY=VX4-VX3
      AAA=DSQRT(UDUMMY*UDUMMY+VDUMMY*VDUMMY)
      DENOM=DSQRT(UX4*UX4+VX4*VX4)
      IF(DENOM.LT.EPSO) GO TO 20
      IF(AAA/DENOM.LT.EPS) GO TO 10
    5 CONV=.FALSE.
      GO TO 100
   10 CONV=.TRUE.
      GO TO 100
   20 IF(AAA.LT.EPSO) GO TO 10
      GO TO 5
  100 RETURN
      END
```

TABLE XIII-B (Continued)

```
          SUBROUTINE HORNER(NA,UA,VA,UX,VX,UB,VB,UPX,VPX)
C     **************************************************************************
C     *                                                                        *
C     * HORNER'S METHOD COMPUTES THE VALUE OF THE POLYNOMIAL P(X) AT A POINT D. *
C     * SYNTHETIC DIVISION IS USED TO DEFLATE THE POLYNOMIAL BY DIVIDING OUT THE *
C     * FACTOR (X-D).                                                          *
C     *                                                                        *
C     **************************************************************************
          DOUBLE PRECISION UX,VX,UPX,VPX,UB,VB,UA,VA
          DIMENSION UA(26),VA(26),UB(26),VB(26)
          UB(1)=UA(1)
          VB(1)=VA(1)
          NUM=NA+1
          DO 10 I=2,NUM
          UB(I)=UA(I)+(UB(I-1)*UX-VB(I-1)*VX)
       10 VB(I)=VA(I)+(VB(I-1)*UX+UB(I-1)*VX)
          UPX=UB(NUM)
          VPX=VB(NUM)
          RETURN
          END
```

TABLE XIII-B (Continued)

```
      SUBROUTINE QUAD(UA,VA,NA,UROOT,VROOT,NROOT,MULTI,EPST)
C     ************************************************************************
C     *                                                                      *
C     * SUBROUTINE QUAD SOLVES DIRECTLY FOR THE ZEROS AND THEIR MULTIPLICITIES *
C     * OF EITHER A QUADRATIC POLYNOMIAL OR A LINEAR FACTOR.  SOLUTION OF THE  *
C     * QUADRATIC IS DONE USING THE QUADRATIC FORMULA.                         *
C     *                                                                      *
C     ************************************************************************
      DOUBLE PRECISION UA,VA,UROOT,VROOT,BBB,UAAA,VAAA,UDISC,VDISC,UDUMM
     1Y,VDUMMY,RDUMMY,SDUMMY,EPST,UBBB,VBBB
      DOUBLE PRECISION DSQRT
      DIMENSION UA(26),VA(26),UROOT(25),VROOT(25),MULTI(25)
      IF(NA.EQ.2) GO TO 7
      IF(NA.EQ.1) GO TO 5
      UROOT(NROOT+1)=0.0
      VROOT(NROOT+1)=0.0
      MULTI(NROOT+1)=1
      NROOT=NROOT+1
      GO TO 50
    5 BBB=UA(1)*UA(1)+VA(1)*VA(1)
      UROOT(NROOT+1)=(-UA(2)*UA(1)-VA(2)*VA(1))/BBB
      VROOT(NROOT+1)=(-VA(2)*UA(1)+UA(2)*VA(1))/BBB
      MULTI(NROOT+1)=1
      NROOT=NROOT+1
      GO TO 50
    7 UDISC=(UA(2)*UA(2)-VA(2)*VA(2))-(4.0*(UA(1)*UA(3)-VA(1)*VA(3)))
      VDISC=(VA(2)*UA(2)+UA(2)*VA(2))-(4.0*(VA(1)*UA(3)+UA(1)*VA(3)))
      BBB=DSQRT(UDISC*UDISC+VDISC*VDISC)
      IF(BBB.LT.EPST) GO TO 10
      CALL COMSQT(UDISC,VDISC,UDUMMY,VDUMMY)
      UBBB=-UA(2)+UDUMMY
      VBBB=-VA(2)+VDUMMY
      RDUMMY=-UA(2)-UDUMMY
      SDUMMY=-VA(2)-VDUMMY
      UAAA=2.0*UA(1)
      VAAA=2.0*VA(1)
      BBB=UAAA*UAAA+VAAA*VAAA
      UROOT(NROOT+1)=(UBBB*UAAA+VBBB*VAAA)/BBB
      VROOT(NROOT+1)=(VBBB*UAAA-UBBB*VAAA)/BBB
      UROOT(NROOT+2)=(RDUMMY*UAAA+SDUMMY*VAAA)/BBB
      VROOT(NROOT+2)=(SDUMMY*UAAA-RDUMMY*VAAA)/BBB
      MULTI(NROOT+1)=1
      MULTI(NROOT+2)=1
      NROOT=NROOT+2
      GO TO 50
   10 UAAA=2.0*UA(1)
      VAAA=2.0*VA(1)
      BBB=UAAA*UAAA+VAAA*VAAA
      UROOT(NROOT+1)=(-UA(2)*UAAA-VA(2)*VAAA)/BBB
      VROOT(NROOT+1)=(-VA(2)*UAAA+UA(2)*VAAA)/BBB
      MULTI(NROOT+1)=2
      NROOT=NROOT+1
   50 RETURN
      END
```

TABLE XIII-B (Continued)

```
      SUBROUTINE COMSQT(UX,VX,UY,VY)
C    ***********************************************************************
C    *                                                                     *
C    * THIS SUBROUTINE COMPUTES THE SQUARE ROOT OF A COMPLEX NUMBER.       *
C    *                                                                     *
C    ***********************************************************************
      DOUBLE PRECISION UX,VX,UY,VY,DUMMY,R,AAA,BBB
      DOUBLE PRECISION DSQRT,DABS
      R=DSQRT(UX*UX+VX*VX)
      AAA=DSQRT(DABS((R+UX)/2.0))
      BBB=DSQRT(DABS((R-UX)/2.0))
      IF(VX) 10,20,30
   10 UY=AAA
      VY=-1.0*BBB
      GO TO 100
   20 IF(UX) 40,50,60
   30 UY=AAA
      VY=BBB
      GO TO 100
   40 DUMMY=DABS(UX)
      UY=0.0
      VY=DSQRT(DUMMY)
      GO TO 100
   50 UY=0.0
      VY=0.0
      GO TO 100
   60 DUMMY=DABS(UX)
      UY=DSQRT(DUMMY)
      VY=0.0
  100 RETURN
      END

$ENTRY
```

APPENDIX D

G.C.D. - NEWTON'S METHOD

1. Use of the Programs

Two programs using the greatest common divisor method supported by Newton's method are presented here. The first is the single precision program. The second program is in double precision and is designed to perform double precision complex arithmetic. These programs are written for use on any computer using FORTRAN IV language. They have been tested on the IBM S/360 mod. 50 computer which has a 32 bit word. However, it may be necessary to change the system functions as described below. The single precision program may be changed to double precision as described below.

After selecting the desired program, the input data should be prepared as described in section 2.

Each program is designed to solve polynomials of degree 25 or less. Both the coefficient of the highest degree term and the constant coefficient should be non-zero. In order to solve polynomials of degree N, where N > 25, certain array dimensions must be changed. These are listed in Table XIV for the main program and subprograms in both single precision and double precision.

TABLE XIV

PROGRAM CHANGES FOR SOLVING POLYNOMIALS OF DEGREE GREATER THAN 25 BY G.C.D. - NEWTON'S METHOD

Single Precision	Double Precision
Main Program	
P(N+1)	UP(N+1),VP(N+1)
OIAPP(N)	UOIAPP(N),VOIAPP(N)
ROOT(N)	UROOT(N),VROOT(N)
MULTI(N)	MULTI(N)
DP(N+1)	UDP(N+1),VDP(N+1)
RK(N+1)	URK(N+1),VRK(N+1)
RIAPP(N)	URIAPP(N),VRIAPP(N)
H(N+1)	UH(N+1),VH(N+1)
DPOLY(N+1)	UDPOLY(N+1),VDPOLY(N+1)

Subroutine METHOD

See main program of Table I in Appendix B.

Subroutines GENAPP, HORNER, BETTER, and QUAD

See corresponding part of Table I in Appendix B.

Single Precision	Double Precision
Subroutine DERIV	
A(N+1)	UA(N+1),VA(N+1)
B(N+1)	UB(N+1),VB(N+1)
Subroutine DIVIDE	
F(N+1)	UF(N+1),VF(N+1)
G(N+1)	UG(N+1),VG(N+1)
H(N+1)	UH(N+1),VH(N+1)
C(N+1)	UC(N+1),VC(N+1)
Subroutine GCD	
R1(N+1)	UR1(N+1),VR1(N+1)
R2(N+1)	UR2(N+1),VR2(N+1)
T(N+1)	UT(N+1),VT(N+1)
RK(N+1)	URK(N+1),VRK(N+1)
P(N+1)	UP(N+1),VP(N+1)
DP(N+1)	UDP(N+1),VDP(N+1)

TABLE XIV (Continued)

Single Precision	Double Precision
Subroutine MULTIP	
A(N+1)	UA(N+1),VA(N+1)
ROOT(N)	UROOT(N),VROOT(N)
WORK(N+1)	UWORK(N+1),VWORK(N+1)
MULTI(N)	MULTI(N)
B(N+1)	UB(N+1),VB(N+1)
C(N+1)	UC(N+1),VC(N+1)

Certain computers may require that the system functions of Table II in Appendix B be changed in the single precision and double precision programs.

When used on the IBM S/360 with the WATFOR compiler for FORTRAN IV, the system functions in Table XIV-A must be typed in a declaration statement. These also appear in the program listing. For use without the WATFOR compiler or on other computers, these system functions might have to be removed. A "c" denotes a comples number and an "r" denotes a real number. For subroutines not listed, see the corresponding subroutine of Table II-A in Appendix B.

TABLE XIV-A

SYSTEM FUNCTIONS IN THE G.C.D. - NEWTON'S METHOD
TO BE TYPED WHEN THE WATFOR COMPILER IS USED

	Single Precision	Double Precision
Subroutines METHOD, GCD, and MULTIP		
Square root -		DSQRT(r)

The single precision program may be converted to double precision for use on machines equipped to perform double precision complex arithmetic provided the following changes or their equivalent are made and the system functions of Table XV are used and typed in a declaration statement where necessary. The changes presented below are those required for the IBM S/360. A "c" denotes a complex number and an "r" denotes a real number. The format statements should be changed from E-type to D-type.

In the main program and each subprogram change COMPLEX $c_1, c_2, \ldots$ to COMPLEX*16 $c_1, c_2, \ldots$ and add IMPLICIT REAL*8(A-H,O-Z).

TABLE XV

SYSTEM FUNCTIONS FOR CONVERTING SINGLE PRECISION G.C.D. - NEWTON'S METHOD TO DOUBLE PRECISION

Single Precision		Double Precision
	Subroutines METHOD, ALTER, NEWTON, CHECK, BETTER, QUAD, GCD, and MULTIP	
CABS(c)	- absolute value -	CDABS(c)
	Subroutines GENAPP, ALTER, and QUAD	
See corresponding part of Table III in Appendix B.		

2. Input Data for G.C.D. - Newton's Method

The input data for the G.C.D. - Newton's method is prepared as described in Appendix B, § 2 except for some of the variable names and one added item to the control card. The control card is described in Table XVI and illustrated in Figure 13. The correspondence of variable names for the coefficient data, initial approximation data, and end card are given in Table XVII.

TABLE XVI

CONTROL DATA FOR G.C.D. - NEWTON'S METHOD

Variable Name	Card Columns	Description
NOPOLY	c.c. 1-2	Number of the polynomial. Integer. Right justified.
NP	c.c. 4-5	Degree of the polynomial. Integer. Right justified.
NIAP	c.c. 7-8	Number of initial approximations to be read. Integer. Right justified. If no initial approximations are given, leave blank.
MAX	c.c. 19-21	Maximum number of iterations. Integer. Right justified. 200 is recommended.
EPSG	c.c. 23-28	Test for zero in subroutine GCD. Real. Right justify. 1.E-02 (1.D-03) is recommended.
EPSCNV	c.c. 30-35	Convergence requirement. Real. Right justify. 1.E-05 (1.D-10) is recommended.
EPSQ	c.c. 37-42	Test for zero in subroutine QUAD. Real. Right justify. 1.E-10 (1.D-20) is recommended.

TABLE XVI (Continued)

Variable Name	Card Columns	Description
EPSMUL	c.c. 44-49	Multiplicity requirement. Real. Right justify. 1.E-01 (1.D-02) is recommended.
XSTART	c.c. 64-70	Magnitude at which to begin generating initial approximations. Real. Right justify. This is a special feature of the program and may be omitted.
XEND	c.c. 72-78	Magnitude at which to end the generating of initial approximations. Real. Right justify. This is a special feature of the program and may be omitted.
KCHECK	c.c. 80	This should be left blank.

TABLE XVII

CORRESPONDENCE OF NEWTON'S AND G.C.D. - NEWTON'S INPUT DATA VARIABLES

Newton's Method	G.C.D. - Newton's Method
Coefficient Card	
A(RA)	P(UP)
A(VA)	P(VP)
Initial Approximation Card	
XZERO(RXZERO)	OIAPP(UOIAPP)
XZERO(VXZERO)	OIAPP(VOIAPP)
End Card	
KCHECK	KCHECK

00	0	00	0	00	0111111111	122	2	222222	2	333333	3	333444	4	444444	55555555556666	6666667	7	7777777	7	8
12	3	45	6	78	9012345678	901	2	345678	9	012345	6	789012	3	456789	01234567890123	4567890	1	2345678	9	0
NOPOLY		NP		NIAP		MAX		EPSG		EPSCNV		EPSQ		EPSMUL		XSTART		XEND		KCHECK
1		7		7		200		1.E-02		1.E-05		1.E-10		1.E-01		1.0E+01		2.0E+01		
1		7		7		200		1.D-03		1.D-10		1.D-20		1.D-02		1.0D+01		2.0D+01		

Figure 13. Control Card for G.C.D. - Newton's Method

3. Variables Used in G.C.D. - Newton's Method

The definitions of the major variables used in the G.C.D. - Newton's method are given in Table XVIII. For definitions of variables not listed in this table see the definitions of variables for the corresponding subroutine in Table VIII of Appendix B. The symbols and notation used here are the same as for Table VIII of Appendix B and are described in Appendix B, § 3.

4. Description of Program Output

The output from the G.C.D. - Newton's method is the same as for Newton's method as described in Appendix B, § 4 in addition to the following. The polynomial containing only simple roots; that is, after all multiple roots have been removed, is printed, coefficient of highest degree term first, under the heading "THE COEFFICIENTS OF H(X) = P(X)/G.C.D. ARE" (Exhibit L). Also the test for zero in subroutine GCD is printed as read from the control card.

5. Informative and Error Messages

The output may contain informative messages printed as an aid to the user. These are:

"THE EPSILON (XXX) CHECK IN SUBROUTINE MULTIP INDICATES THAT ROOT(YY) = ZZZ IS NOT CLOSE ENOUGH TO BE A TRUE ROOT. IT IS PRINTED BELOW WITH MULTIPLICITY 0." XXX is the multiplicity requirement, YY is the number of the root obtained after the attempt to improve accuracy, and ZZZ is the value of the root after the attempt to improve accuracy. This message indicates that the value of the polynomial at the root does not meet the requirement for multiplicities. The root, however, is usually a good approximation to the true root since convergence was obtained both before and after the attempt to improve accuracy.

"NO ZEROS WERE FOUND FOR POLYNOMIAL NUMBER XX." XX is the number of the polynomial. The message indicates that Newton's method did not converge to any root of this polynomial.

For a description of other messages see those of Newton's method given in Appendix B, § 5.

TABLE XVIII

VARIABLES USED IN THE G.C.D. - NEWTON'S METHOD

Single Precision Variable	Type	Double Precision Variable	Type	Disposition of Argument	Description
					Main Program
I01	I	I01	I		Unit number of input device
I02	I	I02	I		Unit number of output device
NOPOLY	I	NOPOLY	I		Number of the polynomial P(X)
NP	I	NP	I		Degree of P(X)
ANAME	A	ANAME	A		Name of method used (NEWTONS)
MAX	I	MAX	I		Maximum number of iterations permitted
NIAP	I	NIAP	I		Number of initial approximations to be read
EPSG	R	EPSG	R		Tolerance check for zero (0) in Subroutine GCD
KCHECK	I	KCHECK	I		Program control - KCHECK = 1 terminates execution
P	C	UP,VP	R		Array of coefficients of original polynomial, (P(X))
OIAPP	C	UOIAPP,VOIAPP	R		Array of initial approximations
NROOT	I	NROOT	I		Number of distinct roots found
IROOT	I	IROOT	I		Number of distinct roots found by the iterative process i.e. not as a result of Subroutine QUAD
EPSCNV	R	EPSCNV	R		Tolerance check for convergence
EPSQ	R	EPSQ	R		Tolerance check for zero (0) in QUAD
EPSMUL	R	EPSMUL	R		Tolerance check for multiplicities
XSTART	R	XSTART	R		Magnitude at which to begin generating approximations (initial)
XEND	R	XEND	R		Magnitude at which to end the generating of initial approximations
ROOT	C	UROOT,VROOT	R		Array of roots found
DP	C	UDP,VDP	R		Array of coefficients of derivative, $P'(X)$, of P(X)
NDP	I	NDP	I		Degree of derivative $P'(X)$

TABLE XVII (Continued)

Single Precision Variable	Type	Double Precision Variable	Type	Disposition of Argument	Description
RK	C	URK,VRK	R		Array of coefficients of the greatest common divisor of P(X) and P'(X)
NRK	I	NRK	I		Degree of g.c.d. of P(X) and P'(X)
H	C	UH,VH	R		Array of coefficients of quotient polynomial i.e. P(X)/g.c.d.
NH	I	NH	I		Degree of quotient polynomial P(X)/g.c.d.
RIAPP	C	URIAPP,VRIAPP	R		Array of approximations (initial or altered) producing convergence
DPOLY	C	UDPOLY,VDPOLY	R		Array of coefficients of deflated polynomial for which no roots were found
MULTI	I	MULTI	I		Array of multiplicities of each root
ND	I	ND	I		Program control and number of coefficients of deflated polynomial for which no zeros were found
				Subroutine GCD	
P	C	UP,VP	R	E	Array of coefficients of P(X)
NP	I	NP	I	E	Degree of P(X)
DP	C	UDP,VDP	R	E	Array of coefficients of P'(X)
NDP	I	NDP	I	E	Degree of P'(X)
R1	C	UR1,VR1	R		Array of coefficients of dividend polynomial
R2	C	UR2,VR2	R		Array of coefficients of divisor polynomial
N1	I	N1	I		Degree of dividend polynomial (R1)
N2	I	N2	I		Degree of divisor polynomial (R2)
T	C	UT,VT	R		Array of coefficients of difference polynomial (R1 - U(R2))
U	C	UU,VU	R		Quotient $(R1_1/R2_1)$
M	I	M	I		Degree of difference polynomial (T)

TABLE XVIII (Continued)

Single Precision Variable	Type	Double Precision Variable	Type	Disposition of Argument	Description
RK	C	URK,VRK	R	R	Array of coefficients of greatest common divisor of P(X) and P'(X)
NK	I	NK	I	R	Degree of g.c.d. of P(X) and P'(X)
EPSG	R	EPSG	R	E	Tolerance check for zero (0)
					Subroutine DERIV
A	C	UA,VA	R	E	Array of coefficients of polynomial, P(X)
NA	I	NA	I	E	Degree of P(X)
B	C	UB,VB	R	R	Array of coefficients of derivative, P'(X)
NB	I	NB	I	R	Degree of P'(X)
					Subroutine MULTIP
A	C	UA,VA	R	E	Array of coefficients of original polynomial, P(X)
NA	I	NA	I	E	Degree of original polynomial, P(X)
ROOT	C	UROOT,VROOT	R	E	Array of roots of P(X)
NR	I	NR	I	E	Number of roots (distinct) in array ROOT
EPSPX	R	EPSPX	R	E	Tolerance check for multiplicities
MULTI	I	MULTI	I	R	Array of multiplicities of each root
WORK	C	UWORK,VWORK	R		Working array for coefficients of current polynomial
NWORK	I	NWORK	I		Degree of polynomial whose coefficients are in WORK i.e. the current polynomial
B	C	UB,VB	R		Array of coefficients of newly deflated polynomial
C	C	UC,VC	R		Array containing sequence of values leading to the derivative
PX	C	UPX,VPX	R		Value of the polynomial at a point
DPX	C	UDPX,VDPX	R		Derivative of the polynomial at a point

TABLE XVIII (Continued)

Single Precision Variable	Single Precision Type	Double Precision Variable	Double Precision Type	Disposition of Argument	Description
KANS	I	KANS	I		KANS = 1 implies convergence; KANS = 0 implies no convergence
IO2	I	IO2	I	C	Unit number of output device
					Subroutine DIVIDE
F	C	UF,VF	R	E	Array of coefficients of dividend polynomial
NF	I	NF	I	E	Degree of dividend polynomial
G	C	UG,VG	R	E	Array of coefficients of divisor polynomial
NG	I	NG	I	E	Degree of divisor polynomial
H	C	UH,VH	R	R	Array of coefficients of quotient polynomial
NH	I	NH	I	R	Degree of quotient polynomial
TERM	C	UTERM,VTERM	R		Dummy variable used for temporary storage
NNF	I	NNF	I		Degree of dividend polynomial to be deflated
X	C	UX,VX	R		Value at which to deflate polynomial F(X)
C	C	UC,VC	R		Array of sequence of values leading to the derivative
PX	C	UPX,VPX	R		Value of polynomial at the point X
DPX	C	UDPX,VDPX	R		Derivative of the polynomial at the point X

MAIN PROGRAM

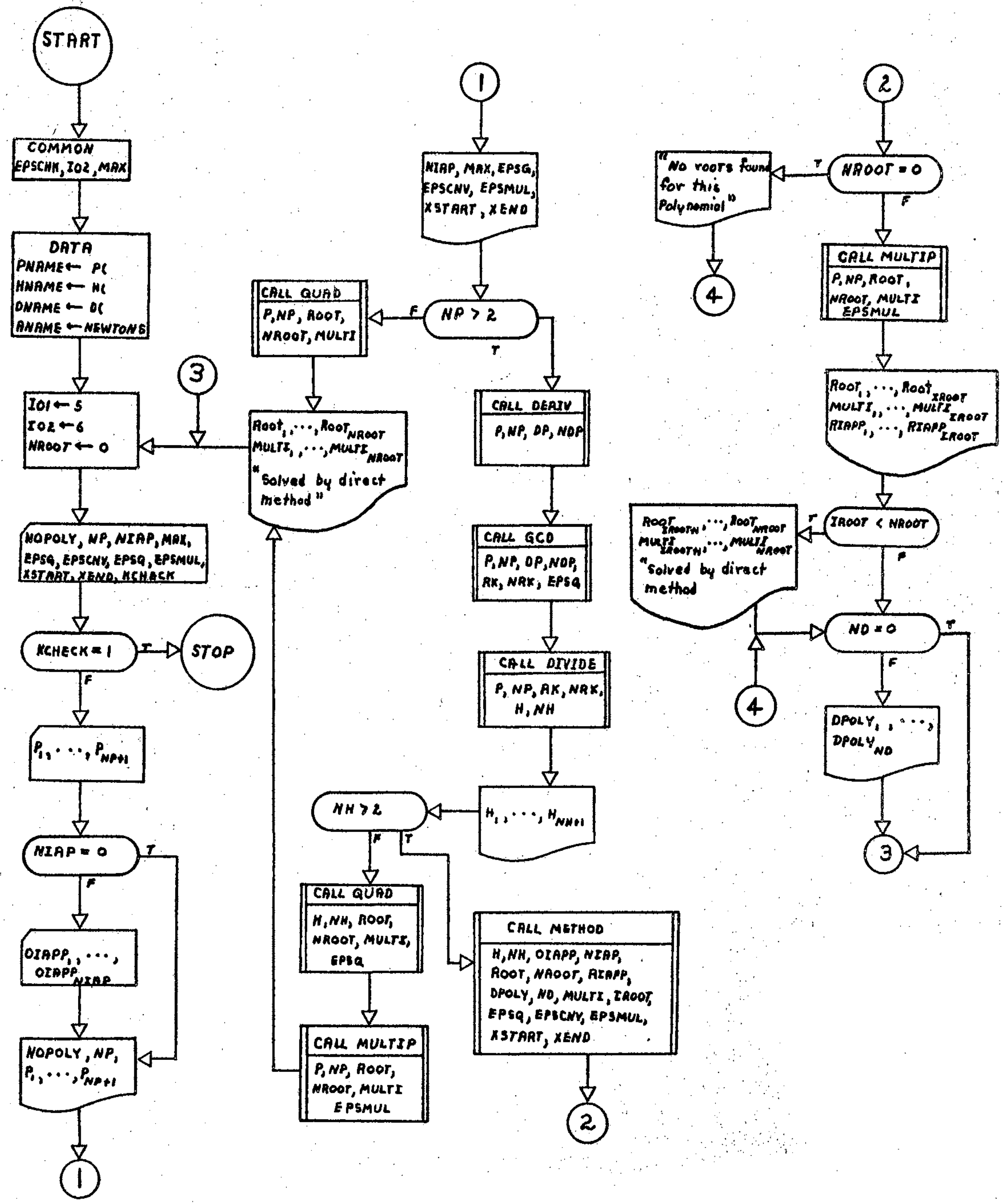

START
COMMON EPSCHK, IO2, MAX
DATA PNAME ← P(HNAME ← H(DNAME ← D(ANAME ← NEWTONS
IO1 ← 5 IO2 ← 6 NROOT ← 0
NOPOLY, NP, NIAP, MAX, EPSG, EPSCNV, EPSQ, EPSMUL, XSTART, XEND, KCHECK
KCHECK = 1
STOP
P1, ···, PNP+1
NIAP = 0
OIAPP1, ···, OIAPPNIAP
NOPOLY, NP, P1, ···, PNP+1
1
3
CALL QUAD P, NP, ROOT, NROOT, MULTI
Root1, ···, RootNROOT MULTI1, ···, MULTINROOT "Solved by direct method"
NH > 2
CALL QUAD H, NH, ROOT, NROOT, MULTI, EPSQ
CALL MULTIP P, NP, ROOT, NROOT, MULTI EPSMUL
1
NIAP, MAX, EPSG, EPSCNV, EPSMUL, XSTART, XEND
NP > 2
CALL DERIV P, NP, DP, NDP
CALL GCD P, NP, DP, NDP, RK, NRK, EPSQ
CALL DIVIDE P, NP, RK, NRK, H, NH
H1, ···, HNH+1
CALL METHOD H, NH, OIAPP, NIAP, ROOT, NROOT, AIAPP, DPOLY, ND, MULTI, IROOT, EPSQ, EPSCNV, EPSMUL, XSTART, XEND
2
2
NROOT = 0
"No roots found for this Polynomial"
4
CALL MULTIP P, NP, ROOT, NROOT, MULTI EPSMUL
Root1, ···, RootIROOT MULTI1, ···, MULTIIROOT AIAPP1, ···, AIAPPIROOT
IROOT < NROOT
RootIROOT+1, ···, RootNROOT MULTIIROOT+1, ···, MULTINROOT "Solved by direct method"
ND = 0
4
DPOLY1, ···, DPOLYND
3
T
F

Figure 13.1. Flow Charts for G.C.D. - Newton's Method

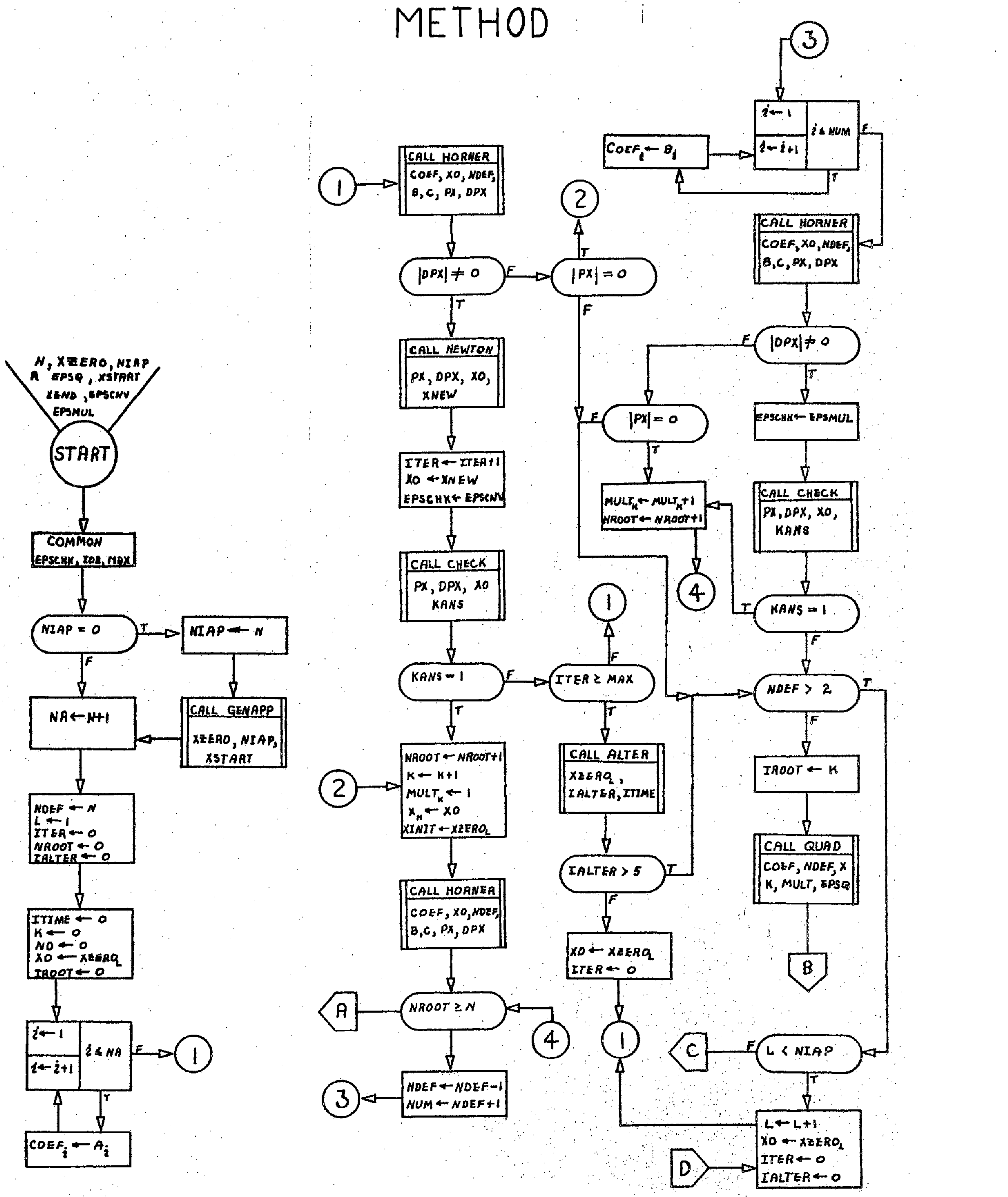


Figure 13.1. (Continued)

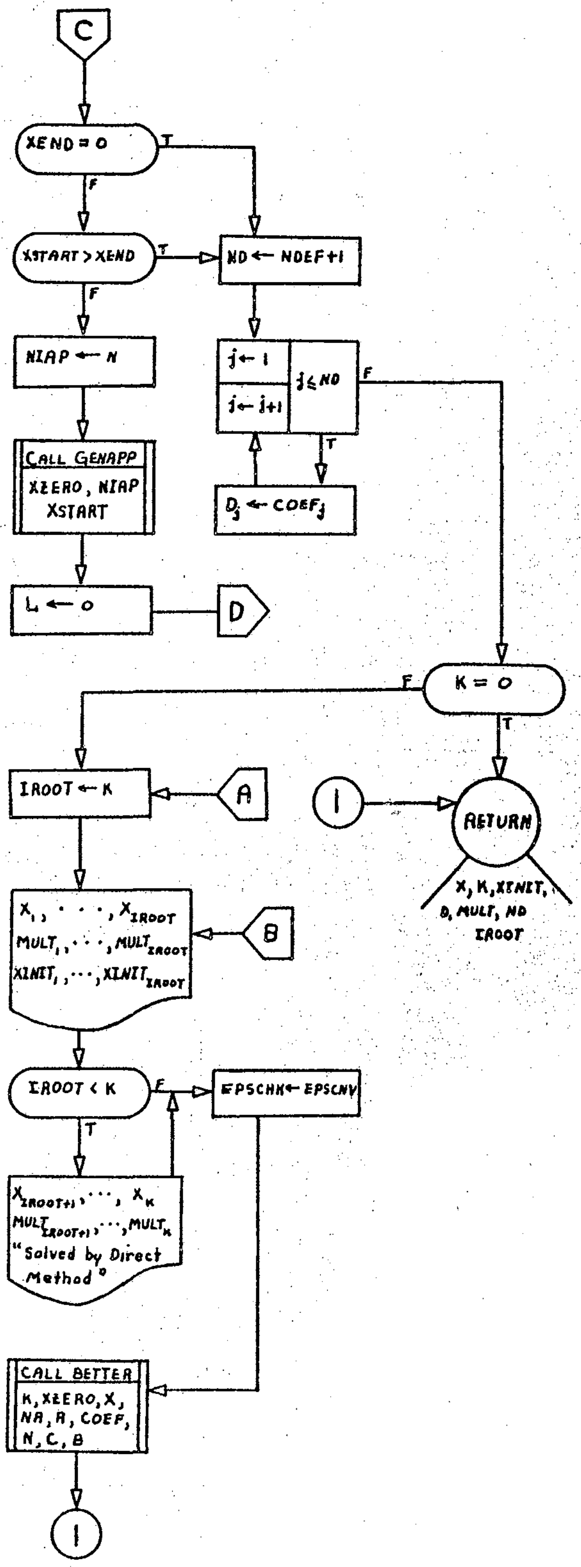
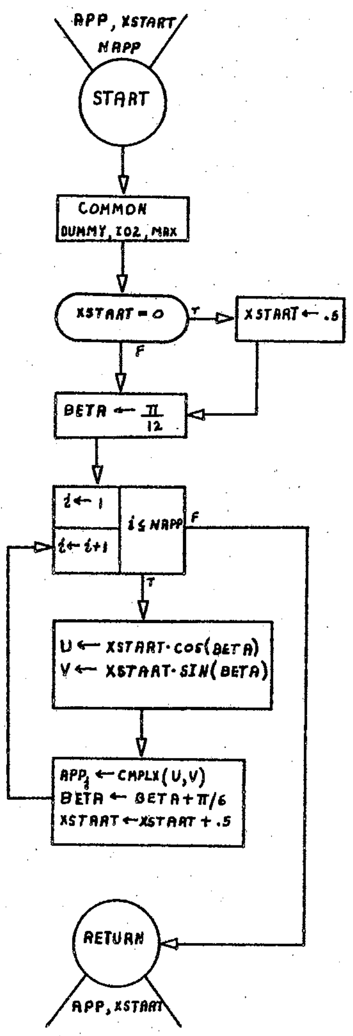


Figure 13.1. (Continued)

GENAPP



ALTER

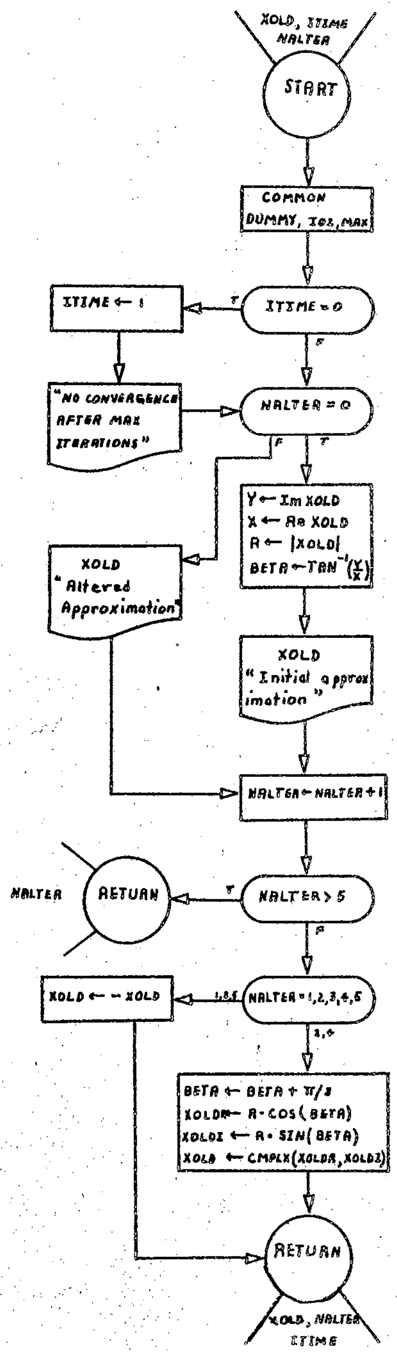
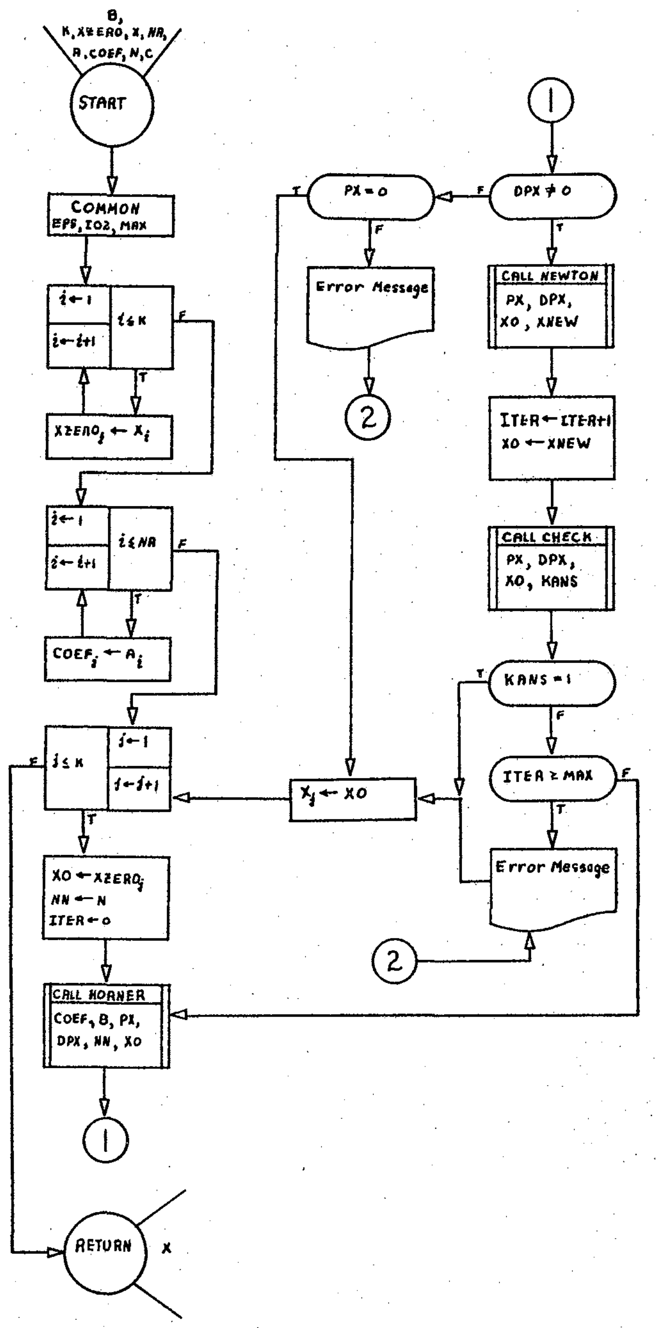


Figure 13.1. (Continued)

BETTER



CHECK

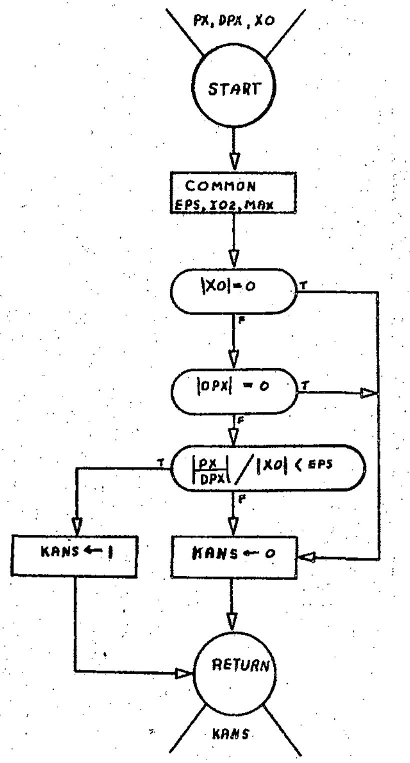
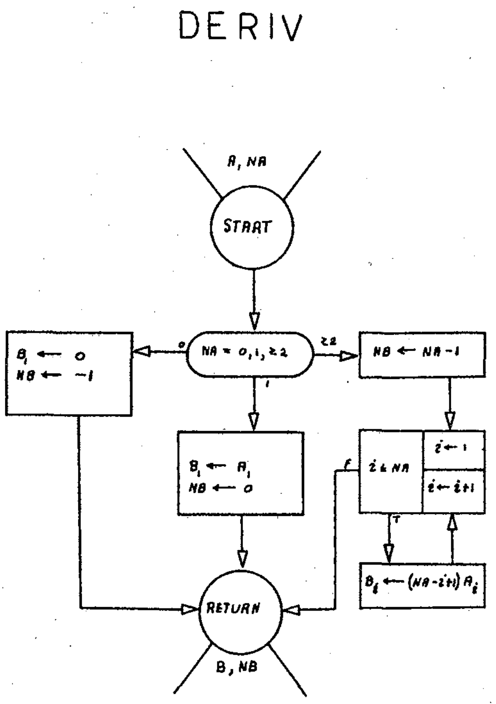


Figure 13.1. (Continued)



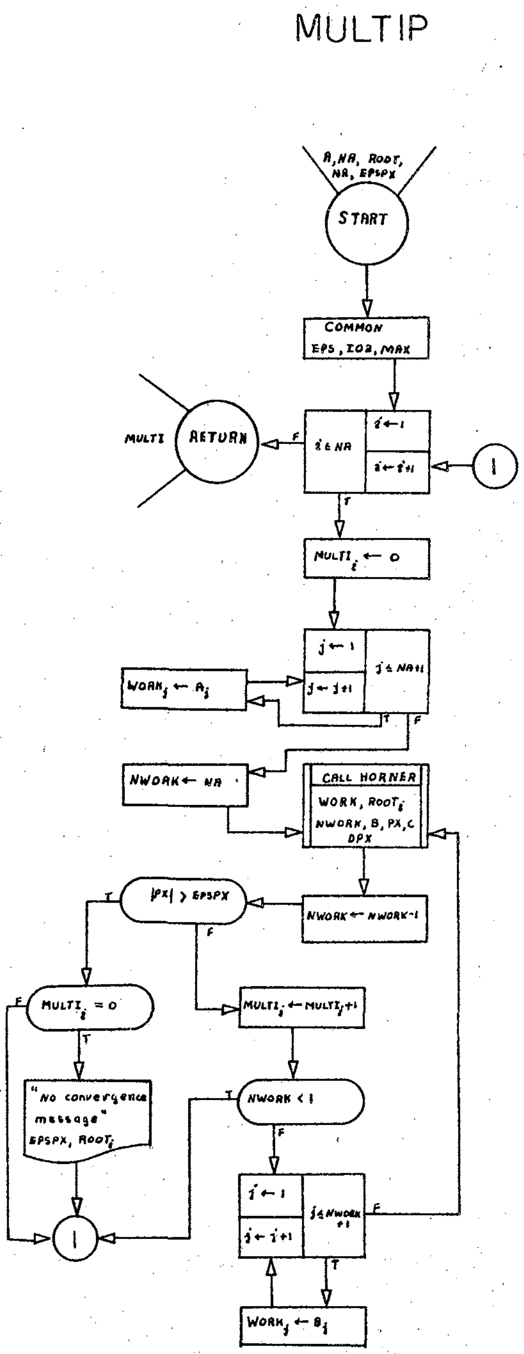


Figure 13.1. (Continued)

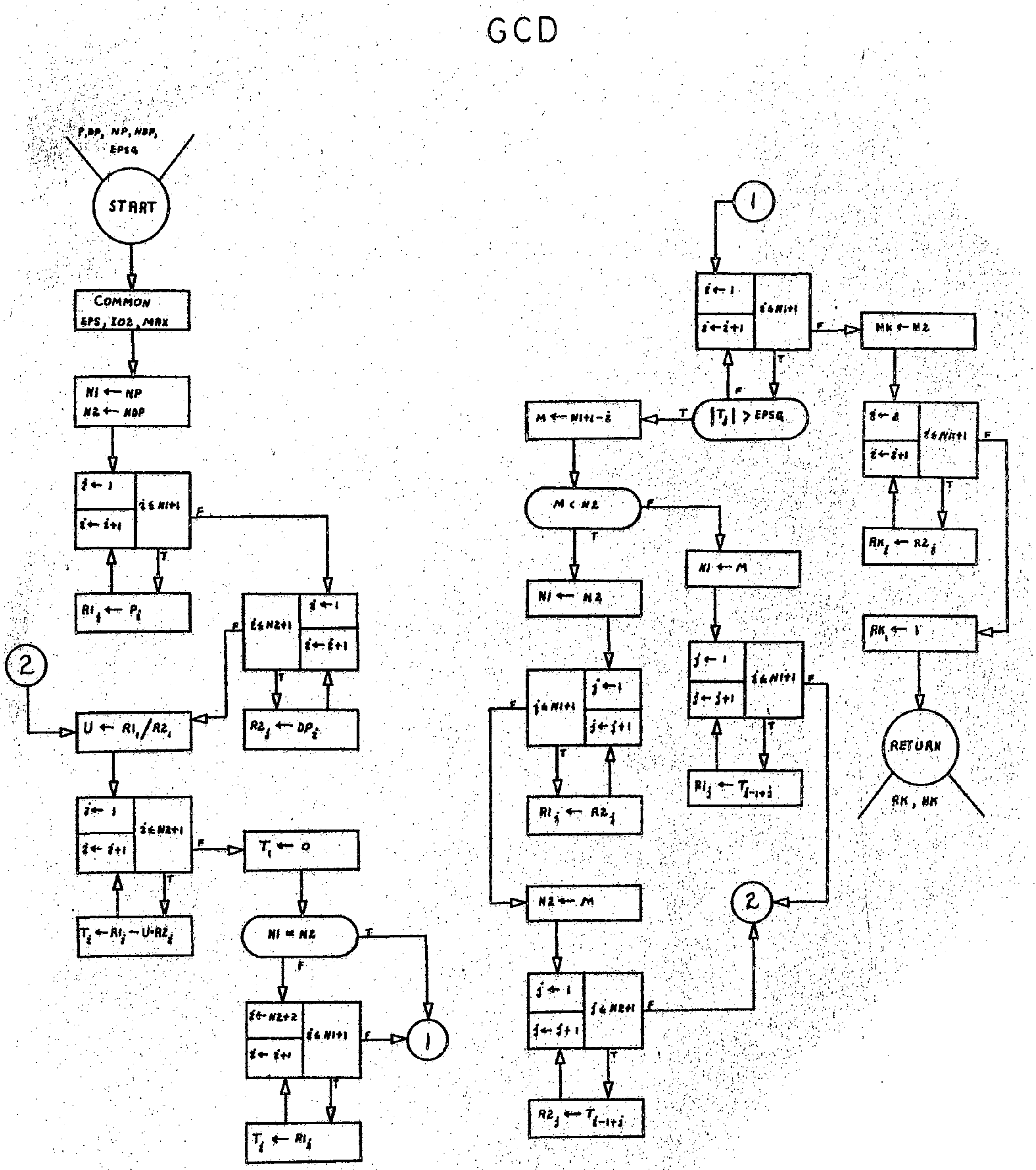


Figure 13.1. (Continued)

DIVIDE

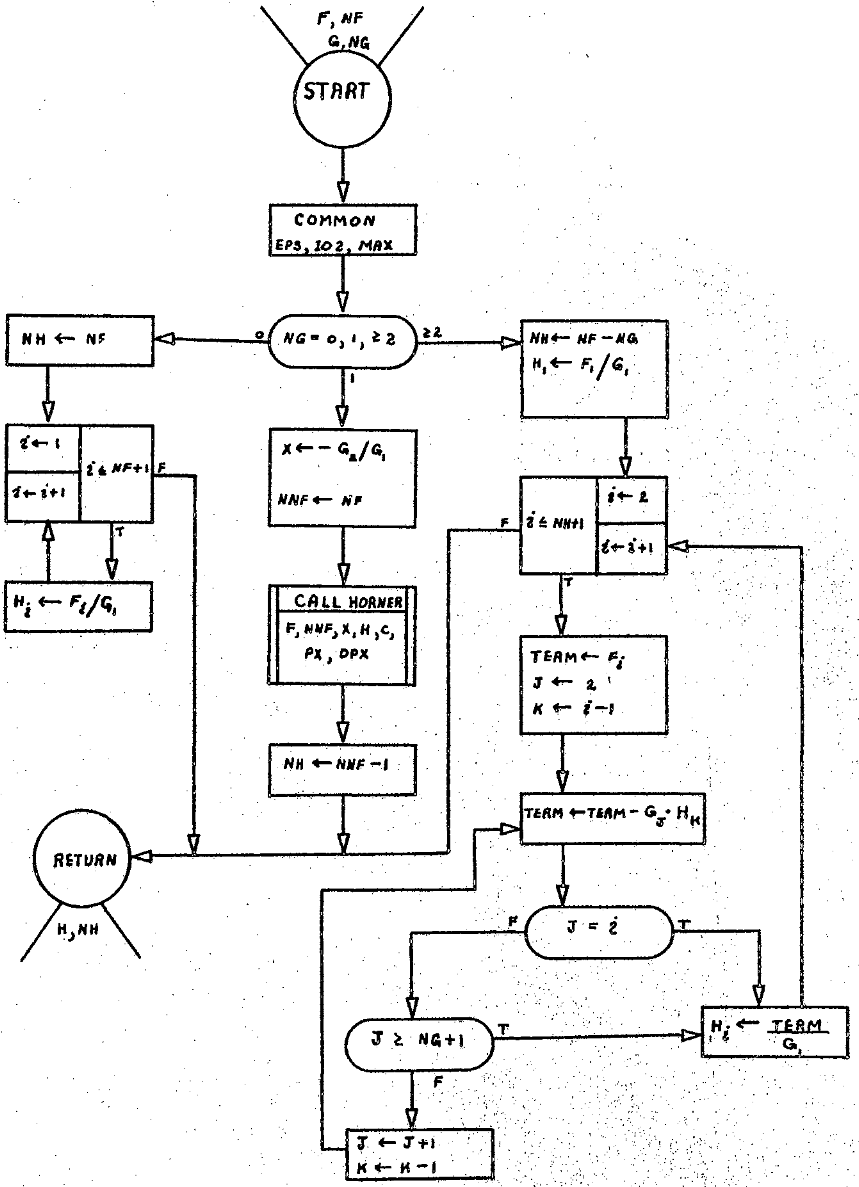
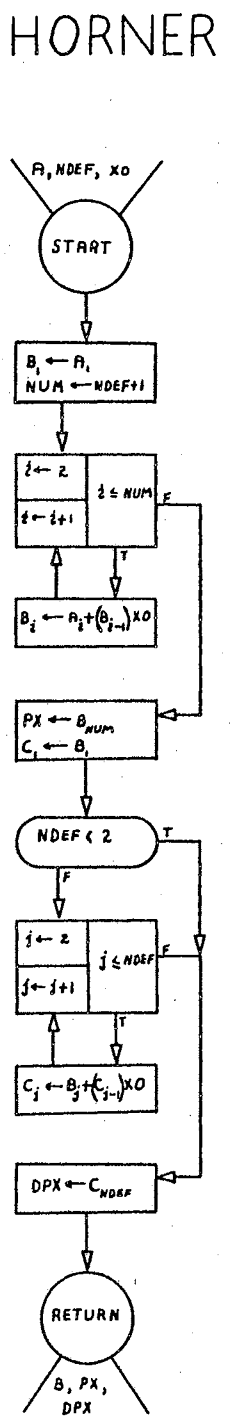


Figure 13.1. (Continued)



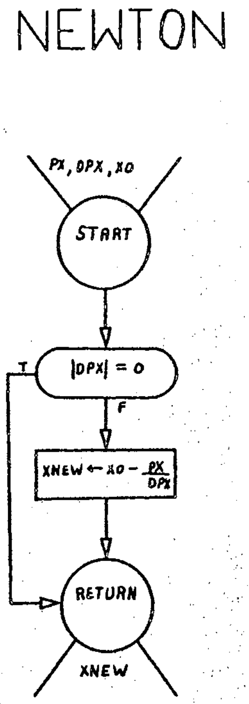
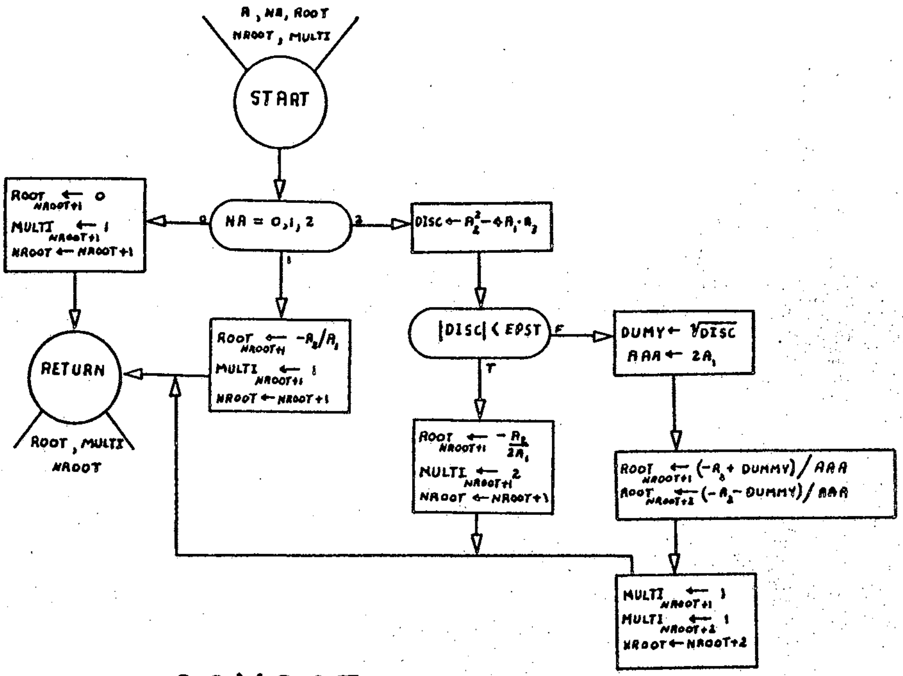


Figure 13.1. (Continued)

QUAD



COMSQT

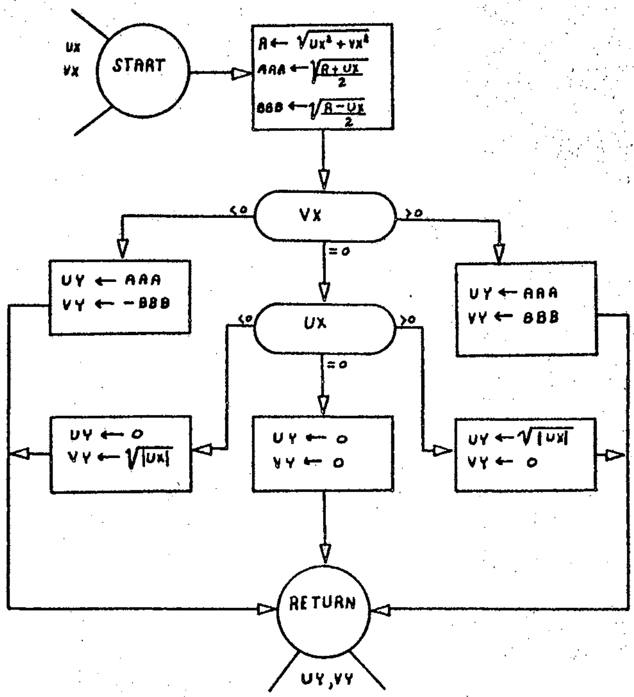


Figure 13.1. (Continued)

TABLE XVIII-A

SINGLE PRECISION PROGRAM FOR G.C.D. - NEWTON'S METHOD

```
$JOB 10414
C     ***************************************************************************
C     *                                                                         *
C     * SINGLE PRECISION PROGRAM FOR G.C.D. - NEWTON'S METHOD                   *
C     *                                                                         *
C     *                                                                         *
C     * THE G.C.D. METHOD EXTRACTS THE ZEROS AND THEIR MULTIPLICITIES OF A      *
C     * POLYNOMIAL OF MAXIMUM DEGREE 25.  ALL MULTIPLE ROOTS ARE REMOVED BY     *
C     * DIVIDING THE POLYNOMIAL BY THE GREATEST COMMON DIVISOR OF THE POLYNOMIAL *
C     * AND ITS DERIVATIVE.  THE ZEROS OF THE RESULTING POLYNOMIAL ARE EXTRACTED.*
C     * AND THEIR MULTIPLICITIES DETERMINED.                                    *
C     *                                                                         *
C     ***************************************************************************
      COMPLEX P,OIAPP,ROOT,DP,RK,RIAPP,H,APPNO,DPOLY
      DIMENSION P(26),OIAPP(25),ROOT(25),MULTI(25),DP(26),RK(26),RIAPP(2
     15),H(26),DPOLY(26),ANAME(2)
      COMMON EPSCHK,IO2,MAX
      DATA PNAME,HNAME,DNAME/2HP(,2HH(,2HD(/
      DATA ANAME(1),ANAME(2)/4HNEWT,4HONS /
      IO1=5
      IO2=6
   10 NROOT=0
      READ(IO1,1000) NOPOLY,NP,NIAP,MAX,EPSG,EPSCNV,EPSQ,EPSMUL,XSTART,X
     1END,KCHECK
      IF(KCHECK.EQ.1) STOP
      KKK=NP+1
      READ(IO1,1010) (P(I),I=1,KKK)
      IF(NIAP.EQ.0) GO TO 20
      READ(IO1,1020) (OIAPP(I),I=1,NIAP)
   20 KKK=NP+1
      WRITE(IO2,1030) ANAME(1),ANAME(2),NOPOLY,NP
      WRITE(IO2,1035) (PNAME,I,P(I),I=1,KKK)
      WRITE(IO2,2060)
      WRITE(IO2,2000) NIAP
      WRITE(IO2,2010) MAX
      WRITE(IO2,2070) EPSG
      WRITE(IO2,2020) EPSCNV
      WRITE(IO2,2030) EPSMUL
      WRITE(IO2,2040) XSTART
      WRITE(IO2,2050) XEND
      IF(NP.GT.2) GO TO 90
      CALL QUAD(P,NP,ROOT,NROOT,MULTI,EPSQ)
   85 WRITE(IO2,1037)
      WRITE(IO2,1086) (I,ROOT(I),MULTI(I),I=1,NROOT)
      GO TO 10
   90 CALL DERIV(P,NP,DP,NDP)
      CALL GCD(P,NP,DP,NDP,RK,NRK,EPSG)
      CALL DIVIDE(P,NP,RK,NRK,H,NH)
      KKK=NH+1
      WRITE(IO2,1060)
      WRITE(IO2,1035) (HNAME,I,H(I),I=1,KKK)
      IF(NH.GT.2) GO TO 150
      CALL QUAD(H,NH,ROOT,NROOT,MULTI,EPSQ)
      CALL MULTIP(P,NP,ROOT,NROOT,MULTI,EPSMUL)
      GO TO 85
  150 CALL METHOD(H,NH,OIAPP,NIAP,ROOT,NROOT,RIAPP,DPOLY,MULTI,ND,IROOT,
     1EPSQ,XSTART,XEND,EPSCNV,EPSMUL)
      IF(NROOT.NE.0) GO TO 170
      WRITE(IO2,1070) NOPOLY
```

TABLE XVIII-A (Continued)

```
      GO TO 200
  170 CALL MULTIP(P,NP,ROOT,NROOT,MULTI,EPSMUL)
      WRITE(IO2,1065)
      WRITE(IO2,1080)
      WRITE(IO2,1085) (I,ROOT(I),MULTI(I),RIAPP(I),I=1,IROOT)
      KKK=IROOT+1
      IF(IROOT.LT.NROOT) WRITE(IO2,1086) (I,ROOT(I),MULTI(I),I=KKK,NROOT
     1)
  200 IF(ND.EQ.0) GO TO 10
  230 WRITE(IO2,1090)
      WRITE(IO2,1035) (DNAME,J,DPOLY(J),J=1,ND)
      GO TO 10
 1080 FORMAT(///1X,13HROOTS OF P(X),37X,14HMULTIPLICITIES,11X,21HINITIAL
     1 APPROXIMATION//)
 1085 FORMAT(2X,5HROOT(,I2,4H) = ,E14.7,3H + ,E14.7,2H I,10X,I2,10X,E14.
     17,3H + ,E14.7,2H I)
 1086 FORMAT(2X,5HROOT(,I2,4H) = ,E14.7,3H + ,E14.7,2H I,10X,I2,11X,23HS
     1OLVED BY DIRECT METHOD)
 1000 FORMAT(3(I2,1X),9X,I3,1X,4(E6.0,1X),13X,2(E7.0,1X),I1)
 1010 FORMAT(2E30.0)
 1020 FORMAT(2E30.0)
 1030 FORMAT(1H1,10X,41HGREATEST COMMON DIVISOR METHOD USED WITH ,2(A4),
     135HMETHOD TO FIND ZEROS OF POLYNOMIALS/11X,18HPOLYNOMIAL NUMBER ,I
     22,11H OF DEGREE ,I2////1X,28HTHE COEFFICIENTS OF P(X) ARE//)
 1035 FORMAT(3X,A2,I2,4H) = ,E14.7,3H + ,E14.7,2H I)
 2000 FORMAT(1X,41HNUMBER OF INITIAL APPROXIMATIONS GIVEN.  ,I2)
 2010 FORMAT(1X,29HMAXIMUM NUMBER OF ITERATIONS.,11X,I3)
 2020 FORMAT(1X,21HTEST FOR CONVERGENCE.,13X,E9.2)
 2030 FORMAT(1X,24HTEST FOR MULTIPLICITIES.,10X,E9.2)
 2040 FORMAT(1X,23HRADIUS TO START SEARCH.,11X,E9.2)
 2050 FORMAT(1X,21HRADIUS TO END SEARCH.,13X,E9.2)
 2060 FORMAT(//1X)
 2070 FORMAT(1X,31HTEST FOR ZERO IN SUBROUTINE GCD,3X,E9.2)
 1037 FORMAT(///,1X,13HZEROS OF P(X),37X,14HMULTIPLICITIES//)
 1060 FORMAT(///1X,42HTHE COEFFICIENTS OF H(X) = P(X)/G.C.D. ARE//)
 1070 FORMAT(///1X,42HNO ZEROS WERE FOUND FOR POLYNOMIAL NUMBER ,I2)
 1065 FORMAT(///1X,61HAFTER  THE ATTEMPT TO IMPROVE ACCURACY, THE ZEROS
     1OF P(X) ARE)
 1090 FORMAT(///,1X,65HCOEFFICIENTS OF DEFLATED POLYNOMIAL FOR WHICH NO
     1ZEROS WERE FOUND//)
      END
```

TABLE XVIII-A (Continued)

```
      SUBROUTINE METHOD(A,N,XZERO,NIAP,X,K,XINIT,D,MULT,ND,IROOT,EPSQ,XS
     1TART,XEND,EPSCNV,EPSMUL)
C     ******************************************************************************
C     *                                                                            *
C     * THIS SUBROUTINE USES NEWTON'S METHOD TO EXTRACT THE ZEROS OF A            *
C     * POLYNOMIAL. A SEQUENCE OF APPROXIMATIONS IS CONSTRUCTED CONVERGING TO A   *
C     * ZERO OF THE POLYNOMIAL BY USING THE ITERATION FORMULA                      *
C     *              X(N+1) = X(N)-P(X(N))/P'(X(N)).                               *
C     *                                                                            *
C     ******************************************************************************
      COMPLEX A,XZERO,B,COEF,X,XINIT,C,D,PX,DPX,XNEW,XO
      DIMENSION A(26),B(26),C(26),D(26),COEF(26),MULT(25),XZERO(25),X(25
     1),XINIT(25)
      COMMON EPSCHK,IO2,MAX
      IF(NIAP.NE.0) GO TO 1
      NIAP=N
      CALL GENAPP(XZERO,NIAP,XSTART)
    1 NA=N+1                                                            130
      NDEF=N
      L=1                                                               152
      ITER=0                                                            154
      NROOT=0                                                           160
      IROOT=0
      IALTER=0                                                          164
      ITIME=0
      ND=0
      K=0                                                               168
      XO=XZERO(L)                                                       180
      DO 5 I=1,NA                                                       188
    5 COEF(I)=A(I)                                                      190
   10 CALL HORNER(COEF,B,PX,DPX,NDEF,XO,C)
      ABPX=CABS(PX)
      ABDPX=CABS(DPX)
      IF(ABDPX.NE.0.0) GO TO 20
      IF(ABPX.EQ.0.0) GO TO 70
      GO TO 110
   20 CALL NEWTON(PX,DPX,XO,XNEW)                                       198
      ITER=ITER+1                                                       200
      XO=XNEW                                                           204
      EPSCHK=EPSCNV
      CALL CHECK(PX,DPX,XO,KANS)
      IF(KANS.EQ.1) GO TO 70                                            194
      IF(ITER.GE.MAX) GO TO 40
      GO TO 10                                                          208
   40 CALL ALTER(XZERO(L),IALTER,ITIME)
      IF(IALTER.GT.5) GO TO 110
      XO=XZERO(L)
      ITER=0                                                            244
      GO TO 10                                                          248
   60 ND=NDEF+1
      DO 65 J=1,ND
   65 D(J)=COEF(J)
      GO TO 140
   70 NROOT=NROOT+1                                                     268
      K=K+1                                                             272
      MULT(K)=1                                                         276
      X(K)=XO                                                           280
      XINIT(K)=XZERO(L)                                                 288
      CALL HORNER(COEF,B,PX,DPX,NDEF,XO,C)
   80 IF(NROOT.GE.N) GO TO 147
```

TABLE XVIII-A (Continued)

```
      NDEF=NDEF-1
      NUM=NDEF+1
      DO 105 I=1,NUM                                                    294
  105 COEF(I)=B(I)                                                      296
      CALL HORNER(COEF,B,PX,DPX,NDEF,XO,C)
      ABPX=CABS(PX)
      ABDPX=CABS(DPX)
      IF(ABDPX.NE.0.0) GO TO 107
      IF(ABPX.EQ.0.0) GO TO 130
      GO TO 110
  107 CONTINUE
      EPSCHK=EPSMUL
      CALL CHECK(PX,DPX,XO,KANS)
      IF(KANS.EQ.1) GO TO 130                                           300
  110 IF(NDEF.GT.2) GO TO 113
      IROOT=K
      CALL QUAD(COEF,NDEF,X,K,MULT,EPSQ)
      GO TO 150
  113 IF(L.LT.NIAP) GO TO 115
      IF(XEND.EQ.0.0) GO TO 60
      IF(XSTART.GT.XEND) GO TO 60
      NIAP=N
      CALL GENAPP(XZERO,NIAP,XSTART)
      L=0
  115 L=L+1
      XO=XZERO(L)                                                       312
      ITER=0                                                            316
      IALTER=0                                                          320
      GO TO 10                                                          324
  130 MULT(K)=MULT(K)+1                                                 328
      NROOT=NROOT+1                                                     332
      GO TO 80                                                          336
  140 IF(K.EQ.0) GO TO 160                                              338
  147 IROOT=K
  150 WRITE(IO2,1025)
      WRITE(IO2,1050)                                                   380
      WRITE(IO2,1060) (I,X(I),MULT(I),XINIT(I),I=1,IROOT)
      KKK=IROOT+1
      IF(IROOT.LT.K) WRITE(IO2,1062) (I,X(I),MULT(I),I=KKK,K)
      EPSCHK=EPSCNV
      CALL BETTER(K,XZERO,X,NA,A,COEF,N,C,B)
  160 RETURN
 1025 FORMAT(///1X,61HBEFORE THE ATTEMPT TO IMPROVE ACCURACY, THE ZEROS
     1OF P(X) ARE)
 1050 FORMAT(///1X,13HZEROS OF P(X),38X,14HMULTIPLICITIES,11X,21HINITIAL
     1 APPROXIMATION/)
 1060 FORMAT(3X,5HROOT(,I2,4H) = ,E14.7,3H + ,E14.7,2H I,9X,I2,12X,E14.7
     1,3H + ,E14.7,2H I)
 1062 FORMAT(3X,5HROOT(,I2,4H) = ,E14.7,3H + ,E14.7,2H I,9X,I2,12X,23HSO
     1LVED BY DIRECT METHOD)
      END                                                               450
```

TABLE XVIII-A (Continued)

```
      SUBROUTINE GENAPP(APP,NAPP,XSTART)
C     ******************************************************************************
C     *                                                                            *
C     * SUBROUTINE GENAPP GENERATES N INITIAL APPROXIMATIONS, WHERE N IS THE       *
C     * DEGREE OF THE ORIGINAL POLYNOMIAL.                                         *
C     *                                                                            *
C     ******************************************************************************
      COMPLEX APP
      COMPLEX CMPLX
      DIMENSION APP(25)
      COMMON DUMMY,IO2,MAX
      IF(XSTART.EQ.0.0) XSTART=0.5
      BETA=0.2617994
      DO 10 I=1,NAPP
      U=XSTART*COS(BETA)
      V=XSTART*SIN(BETA)
      APP(I)=CMPLX(U,V)
      BETA=BETA+0.5235988
   10 XSTART=XSTART+0.5
      RETURN
      END
```

```
      SUBROUTINE ALTER(XOLD,NALTER,ITIME)
C     ******************************************************************************
C     *                                                                            *
C     * SUBROUTINE ALTER ALTERS THE INITIAL APPROXIMATIONS WHICH PRODUCE NO        *
C     * CONVERGENCE TO A ZERO. THIS IS DONE A MAXIMUM OF 5 TIMES FOR EACH ROOT.    *
C     *                                                                            *
C     ******************************************************************************
      COMPLEX XOLD
      COMPLEX CMPLX
      COMMON DUMMY,IO2,MAX
      IF(ITIME.NE.0) GO TO 5
      ITIME=1
      WRITE(IO2,1010) MAX
    5 IF(NALTER.EQ.0) GO TO 10
      WRITE(IO2,1000) XOLD
      GO TO 20
   10 Y=AIMAG(XOLD)
      X=REAL(XOLD)
      R=CABS(XOLD)
      BETA=ATAN2(Y,X)
      WRITE(IO2,1020) XOLD
   20 NALTER=NALTER+1
      IF(NALTER.GT.5) RETURN
      GO TO (30,40,30,40,30),NALTER
   30 XOLD=-XOLD
      GO TO 50
   40 BETA=BETA+1.0471976
      XOLDR=R*COS(BETA)
      XOLDI=R*SIN(BETA)
      XOLD=CMPLX(XOLDR,XOLDI)
   50 RETURN
 1000 FORMAT(1X,E14.7,3H + ,E14.7,2H I,10X,21HALTERED APPROXIMATION)
 1010 FORMAT(///1X,54HNO CONVERGENCE FOR THE FOLLOWING APPROXIMATIONS AF
     1TER ,I3,12H ITERATIONS.//)
 1020 FORMAT(/1X,E14.7,3H + ,E14.7,2H I,10X,21HINITIAL APPROXIMATION)
      END
```

TABLE XVIII-A (Continued)

```
      SUBROUTINE HORNER(A,B,PX,DPX,NDEF,XO,C)
C   ***********************************************************************
C   *                                                                     *
C   * HORNER'S METHOD COMPUTES THE VALUE OF A POLYNOMIAL P(X) AT A POINT D AND *
C   * ITS DERIVATIVE AT D.  SYNTHETIC DIVISION IS USED TO DEFLATE THE     *
C   * POLYNOMIAL BY DIVIDING OUT THE FACTOR (X-D).                        *
C   *                                                                     *
C   ***********************************************************************
      COMPLEX A,B,PX,DPX,XO,C                                          504
      DIMENSION A(26),B(26),C(26)
      COMMON EPS,IO2,MAX
      B(1)=A(1)                                                        516
      NUM=NDEF+1                                                       520
      DO 10 I=2,NUM                                                    524
   10 B(I)=A(I)+B(I-1)*XO                                              528
      PX=B(NUM)                                                        532
      C(1)=B(1)                                                        540
      IF(NDEF.LT.2) GO TO 25
      DO 20 J=2,NDEF                                                   544
   20 C(J)=B(J)+C(J-1)*XO                                              548
   25 DPX=C(NDEF)
      RETURN                                                           572
      END                                                              580
```

```
      SUBROUTINE NEWTON(PX,DPX,XO,XNEW)                                600
C   ***********************************************************************
C   *                                                                     *
C   * THIS SUBROUTINE CALCULATES A NEW APPROXIMATION FROM THE OLD APPROX- *
C   * IMATION BY USING THE ITERATION FORMULA                              *
C   *              X(N+1) = X(N)-P(X(N))/P'(X(N)).                        *
C   *                                                                     *
C   ***********************************************************************
      COMPLEX PX,DPX,XO,XNEW                                           604
      DDD=CABS(DPX)
      IF(DDD.EQ.0.0) RETURN
      XNEW=XO-(PX/DPX)                                                 612
      RETURN                                                           616
      END                                                              620
```

TABLE XVIII-A (Continued)

```
          SUBROUTINE CHECK(PX,DPX,XO,KANS)
     C   *************************************************************************
     C   *                                                                       *
     C   * THIS SUBROUTINE CHECKS FOR CONVERGENCE OF THE SEQUENCE OF APPROX-     *
     C   * IMATIONS BY TESTING THE EXPRESSION                                    *
     C   * ABSOLUTE VALUE OF (P(X(N))/P'(X(N)))/ABSOLUTE VALUE OF X(N+1).        *
     C   * WHEN IT IS AS SMALL AS DESIRED, CONVERGENCE IS OBTAINED.              *
     C   *                                                                       *
     C   *************************************************************************
          COMPLEX PX,DPX,XO                                                   748
          COMMON EPS,IO2,MAX
          IF(CABS(XO).EQ.0.) GO TO 25
          DDD=CABS(DPX)
          IF(DDD.EQ.0.0) GO TO 25
          IF (CABS(PX/DPX)/CABS(XO).LT.EPS) GO TO 10
          KANS=0                                                              760
          RETURN                                                              764
       10 KANS=1                                                              768
          RETURN                                                              772
       25 KANS=0
          RETURN
          END                                                                 780
```

TABLE XVIII-A (Continued)

```
      SUBROUTINE BETTER(K,XZERO,X,NA,A,COEF,N,C,B)
C   *****************************************************************************
C   *                                                                           *
C   * SUBROUTINE BETTER ATTEMPTS TO IMPROVE THE ACCURACY OF THE ZEROS FOUND     *
C   * BY USING THEM AS INITIAL APPROXIMATIONS WITH NEWTON'S METHOD APPLIED TO   *
C   * THE FULL, UNDEFLATED POLYNOMIAL.                                          *
C   *                                                                           *
C   *****************************************************************************
      COMPLEX XZERO,X,A,COEF,C,B,XO,PX,DPX,XNEW                               804
      DIMENSION XZERO(25),X(25),A(26),COEF(26),C(26),B(26)
      COMMON EPS,IO2,MAX
      DO 10 I=1,K                                                             812
   10 XZERO(I)=X(I)                                                           815
      DO 20 I=1,NA                                                            820
   20 COEF(I)=A(I)                                                            824
      DO 50 J=1,K                                                             828
      XO=XZERO(J)                                                             832
      NN=N                                                                    834
      ITER=0                                                                  836
   30 CALL HORNER(COEF,B,PX,DPX,NN,XO,C)
      ABPX=CABS(PX)
      ABDPX=CABS(DPX)
      IF(ABDPX.NE.0.0) GO TO 33
      IF(ABPX.EQ.0.0) GO TO 40
      GO TO 34
   33 CALL NEWTON(PX,DPX,XO,XNEW)
      ITER=ITER+1                                                             856
      XO=XNEW                                                                 860
      CALL CHECK(PX,DPX,XO,KANS)
      IF(KANS.EQ.1) GO TO 40                                                  844
      IF(ITER.GE.MAX) GO TO 35
      GO TO 30                                                                864
   34 WRITE(IO2,1112) XO
   35 WRITE(IO2,100) J,XZERO(J)
      WRITE(IO2,200) MAX
   40 X(J)=XO                                                                 868
   50 CONTINUE                                                                872
      RETURN                                                                  876
 1112 FORMAT(1HO,36HTHE VALUE OF THE DERIVATIVE AT XO = ,E14.7,3H + ,E14
     1.7,2H I,10H  IS ZERO.)
  100 FORMAT(42HOIN THE ATTEMPT TO IMPROVE ACCURACY, ROOT(,I2,4H) = ,E14
     1.7,3H + ,E14.7,2H I,18H DID NOT CONVERGE.)
  200 FORMAT(33H THE PRESENT APPROXIMATION AFTER ,I3,29H ITERATIONS IS P
     1RINTED BELOW.)
      END                                                                     880
```

TABLE XVIII-A (Continued)

```
      SUBROUTINE DERIV(A,NA,B,NB)
C    ******************************************************************************
C    *                                                                            *
C    * GIVEN A POLYNOMIAL P(X), SUBROUTINE DERIV COMPUTES THE COEFFICIENTS OF    *
C    * ITS DERIVATIVE P'(X).                                                      *
C    *                                                                            *
C    ******************************************************************************
      COMPLEX A,B
      DIMENSION A(26),B(26)
      IF(NA.GE.2) GO TO 30
      IF(NA.EQ.1) GO TO 20
      B(1)=0.0
      NB=-1
      GO TO 50
   20 B(1)=A(1)
      NB=0
      GO TO 50
   30 NB=NA-1
      DO 40 I=1,NA
      BBB=NA-I+1
   40 B(I)=BBB*A(I)
   50 RETURN
      END
```

```
      SUBROUTINE DIVIDE(F,NF,G,NG,H,NH)
C    ******************************************************************************
C    *                                                                            *
C    * GIVEN TWO POLYNOMIALS F(X) AND G(X), SUBROUTINE DIVIDE COMPUTES THE        *
C    * QUOTIENT POLYNOMIAL  H(X) = F(X)/G(X).                                     *
C    *                                                                            *
C    ******************************************************************************
      COMPLEX F,G,H,X,C,PX,DPX,TERM
      DIMENSION F(26),G(26),H(26),C(26)
      COMMON EPS,IO2,MAX
      IF(NG.GE.2) GO TO 60
      IF(NG.EQ.1) GO TO 30
      NH=NF
      KKK=NF+1
      DO 20 I=1,KKK
   20 H(I)=F(I)/G(1)
      GO TO 100
   30 X=-G(2)/G(1)
      NNF=NF
      CALL HORNER(F,H,PX,DPX,NNF,X,C)
      NH=NNF-1
      GO TO 100
   60 NH=NF-NG
      H(1)=F(1)/G(1)
      KKK=NH+1
      DO 90 I=2,KKK
      TERM=F(I)
      J=2
      K=I-1
   70 TERM=TERM-G(J)*H(K)
      IF(J.EQ.I) GO TO 90
      IF(J.GE.NG+1) GO TO 90
      J=J+1
      K=K-1
      GO TO 70
   90 H(I)=TERM/G(1)
  100 RETURN
      END
```

TABLE XVIII-A (Continued)

```
      SUBROUTINE QUAD(A,NA,ROOT,NROOT,MULTI,EPST)
C   ****************************************************************************
C   *                                                                          *
C   * SUBROUTINE QUAD SOLVES DIRECTLY FOR THE ZEROS AND THEIR MULTIPLICITIES   *
C   * OF EITHER A QUADRATIC POLYNOMIAL OR A LINEAR FACTOR.  SOLUTION OF THE    *
C   * QUADRATIC IS DONE USING THE QUADRATIC FORMULA.                           *
C   *                                                                          *
C   ****************************************************************************
      COMPLEX A,DISC,ROOT,DUMMY,AAA
      COMPLEX CSQRT
      DIMENSION A(26),ROOT(25),MULTI(25)
      IF(NA.EQ.2) GO TO 7
      IF(NA.EQ.1) GO TO 5
      ROOT(NROOT+1)=0.0
      MULTI(NROOT+1)=1
      NROOT=NROOT+1
      GO TO 50
    5 ROOT(NROOT+1)=-A(2)/A(1)
      MULTI(NROOT+1)=1
      NROOT=NROOT+1
      GO TO 50
    7 DISC=A(2)*A(2)-(4.0*A(1)*A(3))
      BBB=CABS(DISC)
      IF(BBB.LT.EPST) GO TO 10
      DUMMY=CSQRT(DISC)
      AAA=2.0*A(1)
      ROOT(NROOT+1)=(-A(2)+DUMMY)/AAA
      ROOT(NROOT+2)=(-A(2)-DUMMY)/AAA
      MULTI(NROOT+1)=1
      MULTI(NROOT+2)=1
      NROOT=NROOT+2
      GO TO 50
   10 ROOT(NROOT+1)=(-A(2))/(2.0*A(1))
      MULTI(NROOT+1)=2
      NROOT=NROOT+1
   50 RETURN
      END
```

TABLE XVIII-A (Continued)

```
      SUBROUTINE GCD(P,NP,DP,NDP,RK,NK,EPSG)
C     ***********************************************************************
C     *                                                                     *
C     * GIVEN POLYNOMIALS P(X) AND DP(X) WHERE DEG. DP(X) IS LESS THAN DEG. *
C     * P(X), SUBROUTINE GCD COMPUTES THE GREATEST COMMON DIVISOR OF P(X) AND *
C     * DP(X).                                                              *
C     *                                                                     *
C     ***********************************************************************
      COMPLEX R1,R2,RK,U,T,P,DP
      DIMENSION R1(26),R2(26),T(26),RK(26),P(26),DP(26)
      COMMON EPS,IO2,MAX
      N1=NP
      N2=NDP
      KKK=N1+1
      DO 10 I=1,KKK
   10 R1(I)=P(I)
      KKK=N2+1
      DO 20 I=1,KKK
   20 R2(I)=DP(I)
   50 U=R1(1)/R2(1)
      KKK=N2+1
      DO 70 I=1,KKK
   70 T(I)=R1(I)-U*R2(I)
      T(1)=0.0
      IF(N1.EQ.N2) GO TO 90
      KKK=N1+1
      NNN=N2+2
      DO 80 I=NNN,KKK
   80 T(I)=R1(I)
   90 KKK=N1+1
      DO 100 I=1,KKK
      BBB=CABS(T(I))
      IF(BBB.GT.EPSG) GO TO 120
  100 CONTINUE
      NK=N2
      KKK=NK+1
      DO 110 I=2,KKK
  110 RK(I)=R2(I)/R2(1)
      RK(1)=1.0
      GO TO 200
  120 M=N1+1-I
      IF(M.LT.N2) GO TO 160
      N1=M
      KKK=N1+1
      NNN=I-1
      DO 150 J=1,KKK
  150 R1(J)=T(NNN+J)
      GO TO 50
  160 N1=N2
      KKK=N1+1
      DO 170 J=1,KKK
  170 R1(J)=R2(J)
      N2=M
      KKK=N2+1
      NNN=I-1
      DO 180 J=1,KKK
  180 R2(J)=T(NNN+J)
      GO TO 50
  200 RETURN
      END
```

TABLE XVIII-A (Continued)

```
      SUBROUTINE MULTIP(A,NA,ROOT,NR,MULTI,EPSPX)
C     ***********************************************************************
C     *                                                                     *
C     * GIVEN NR ZEROS OF A POLYNOMIAL, SUBROUTINE MULTIP COMPUTES THEIR    *
C     * MULTIPLICITIES.                                                     *
C     *                                                                     *
C     ***********************************************************************
      COMPLEX A,ROOT,WORK,C,PX,DPX,B
      DIMENSION A(26),ROOT(25),WORK(26),MULTI(25),B(26),C(26)
      COMMON EPS,IO2,MAX
      DO 50 I=1,NR
      MULTI(I)=0
      KKK=NA+1
      DO 20 J=1,KKK
   20 WORK(J)=A(J)
      NWORK=NA
   25 CALL HORNER(WORK,B,PX,DPX,NWORK,ROOT(I),C)
      NWORK=NWORK-1
      BBB=CABS(PX)
      IF(BBB.GT.EPSPX) GO TO 45
      MULTI(I)=MULTI(I)+1
      IF(NWORK.LT.1) GO TO 50
      KKK=NWORK+1
      DO 40 J=1,KKK
   40 WORK(J)=B(J)
      GO TO 25
   45 IF(MULTI(I).EQ.0) WRITE(IO2,1000) EPSPX,I,ROOT(I)
   50 CONTINUE
  100 RETURN
 1000 FORMAT(///14H THE EPSILON (,E14.7,49H) CHECK IN SUBROUTINE MULTIP
     1INDICATES THAT ROOT(,I2,4H) = ,E14.7,3H + ,E14.7,2H I/80H IS NOT C
     2LOSE ENOUGH TO BE A TRUE ROOT.  IT IS PRINTED BELOW WITH MULTIPLIC
     3ITY 0//)
      END

$ENTRY
```

TABLE XVIII-B

DOUBLE PRECISION PROGRAM FOR G.C.D. - NEWTON'S METHOD

```
$JOB 10414
C     ************************************************************************
C     *                                                                      *
C     * DOUBLE PRECISION PROGRAM FOR G.C.D. - NEWTON'S METHOD                *
C     *                                                                      *
C     *                                                                      *
C     * THE G.C.D. METHOD EXTRACTS THE ZEROS AND THEIR MULTIPLICITIES OF A   *
C     * POLYNOMIAL OF MAXIMUM DEGREE 25.  ALL MULTIPLE ROOTS ARE REMOVED BY  *
C     * DIVIDING THE POLYNOMIAL BY THE GREATEST COMMON DIVISOR OF THE POLYNOMIAL *
C     * AND ITS DERIVATIVE.  THE ZEROS OF THE RESULTING POLYNOMIAL ARE EXTRACTED *
C     * AND THEIR MULTIPLICITIES DETERMINED.                                 *
C     *                                                                      *
C     ************************************************************************
      DOUBLE PRECISION UP,VP,UOIAPP,VOIAPP,UROOT,VROOT,UDP,VDP,URK,VRK,U
     1RIAPP,VRIAPP,UH,VH,UDPOLY,VDPOLY,EPSCHK,UDUMMY,VDUMMY,EPSQ,EPSMUL,
     2EPSG,EPSCNV,XSTART,XEND
      DIMENSION ANAME(2),UP(26),VP(26),UOIAPP(25),VOIAPP(25),UROOT(25),V
     1ROOT(25),UDP(26),VDP(26),URK(26),VRK(26),URIAPP(25),VRIAPP(25),UH(
     226),VH(26),UDPOLY(26),VDPOLY(26),MULTI(25)
      COMMON EPSCHK,IO2,MAX
      DATA PNAME,HNAME,DNAME/2HP(,2HH(,2HD(/
      DATA ANAME(1),ANAME(2)/4HNEWT,4HONS /
      IO1=5
      IO2=6
   10 NROOT=0
      READ(IO1,1000) NOPOLY,NP,NIAP,MAX,EPSG,EPSCNV,EPSQ,EPSMUL,XSTART,X
     1END,KCHECK
      IF(KCHECK.EQ.1) STOP
      KKK=NP+1
      READ(IO1,1010) (UP(I),VP(I),I=1,KKK)
      IF(NIAP.EQ.0) GO TO 20
      READ(IO1,1020) (UOIAPP(I),VOIAPP(I),I=1,NIAP)
   20 KKK=NP+1
      WRITE(IO2,1030) ANAME(1),ANAME(2),NOPOLY,NP
      WRITE(IO2,1035) (PNAME,I,UP(I),VP(I),I=1,KKK)
      WRITE(IO2,2060)
      WRITE(IO2,2000) NIAP
      WRITE(IO2,2010) MAX
      WRITE(IO2,2070) EPSG
      WRITE(IO2,2020) EPSCNV
      WRITE(IO2,2030) EPSMUL
      WRITE(IO2,2040) XSTART
      WRITE(IO2,2050) XEND
      IF(NP.GT.2) GO TO 90
      CALL QUAD(UP,VP,NP,UROOT,VROOT,MULTI,EPSQ)
   85 WRITE(IO2,1037)
      WRITE(IO2,1086) (I,UROOT(I),VROOT(I),MULTI(I),I=1,NROOT)
      GO TO 10
   90 CALL DERIV(UP,VP,NP,UDP,VDP,NDP)
      CALL GCD(UP,VP,NP,UDP,VDP,NDP,URK,VRK,NRK,EPSG)
      CALL DIVIDE(UP,VP,NP,URK,VRK,NRK,UH,VH,NH)
      KKK=NH+1
      WRITE(IO2,1060)
      WRITE(IO2,1035) (HNAME,I,UH(I),VH(I),I=1,KKK)
      IF(NH.GT.2) GO TO 150
      CALL QUAD(UH,VH,NH,UROOT,VROOT,NROOT,MULTI,EPSQ)
      CALL MULTIP(UP,VP,NP,UROOT,VROOT,NROOT,MULTI,EPSMUL)
      GO TO 85
  150 CALL METHOD(UH,VH,NH,UOIAPP,VOIAPP,NIAP,UROOT,VROOT,NROOT,URIAPP,V
```

TABLE XVIII-B (Continued)

```
     1RIAPP,UDPOLY,VDPOLY,MULTI,ND,IROOT,EPSQ,XSTART,XEND,EPSCNV,EPSMUL)
      IF(NROOT.NE.0) GO TO 170
      WRITE(IO2,1070) NOPOLY
      GO TO 200
  170 CALL MULTIP(UP,VP,NP,UROOT,VROOT,NROOT,MULTI,EPSMUL)
      WRITE(IO2,1065)
      WRITE(IO2,1080)
      WRITE(IO2,1085) (I,UROOT(I),VROOT(I),MULTI(I),URIAPP(I),VRIAPP(I),
     1I=1,IROOT)
      KKK=IROOT+1
      IF(IROOT.LT.NROOT) WRITE(IO2,1086) (I,UROOT(I),VROOT(I),MULTI(I),I
     1=KKK,NROOT)
  200 IF(ND.EQ.0) GO TO 10
  230 WRITE(IO2,1090)
      WRITE(IO2,1035) (DNAME,J,UDPOLY(J),VDPOLY(J),J=1,ND)
      GO TO 10
 1000 FORMAT(3(I2,1X),9X,I3,1X,4(D6.0,1X),13X,2(D7.0,1X),I1)
 1010 FORMAT(2D30.0)
 1020 FORMAT(2D30.0)
 1030 FORMAT(1H1,10X,41HGREATEST COMMON DIVISOR METHOD USED WITH ,2(A4),
     135HMETHOD TO FIND ZEROS OF POLYNOMIALS/11X,18HPOLYNOMIAL NUMBER ,I
     22,11H OF DEGREE ,I2////1X,28HTHE COEFFICIENTS OF P(X) ARE//)
 1035 FORMAT(3X,A2,I2,4H) = ,D23.16,3H + ,D23.16,2H I)
 1037 FORMAT(///1X,13HZEROS OF P(X),55X,14HMULTIPLICITIES//)
 1086 FORMAT(3X,5HROOT(,I2,4H) = ,D23.16,3H + ,D23.16,2H I,7X,I2,8X,23HS
     1OLVED BY DIRECT METHOD)
 1060 FORMAT(///1X,42HTHE COEFFICIENTS OF H(X) = P(X)/G.C.D. ARE//)
 1070 FORMAT(///1X,42HNO ZEROS WERE FOUND FOR POLYNOMIAL NUMBER ,I2)
 1065 FORMAT(///1X,61HAFTER  THE ATTEMPT TO IMPROVE ACCURACY, THE ZEROS
     1OF P(X) ARE)
 1085 FORMAT(3X,5HROOT(,I2,4H) = ,D23.16,3H + ,D23.16,2H I,7X,I2,7X,D23.
     116,3H + ,D23.16,2H I)
 1090 FORMAT(///,1X,65HCOEFFICIENTS OF DEFLATED POLYNOMIAL FOR WHICH NO
     1ZEROS WERE FOUND//)
 1080 FORMAT(///1X,13HROOTS OF P(X),52X,14HMULTIPLICITIES,17X,21HINITIAL
     1 APPROXIMATION//)
 2000 FORMAT(1X,41HNUMBER OF INITIAL APPROXIMATIONS GIVEN.  ,I2)
 2010 FORMAT(1X,29HMAXIMUM NUMBER OF ITERATIONS.,11X,I3)
 2020 FORMAT(1X,21HTEST FOR CONVERGENCE.,13X,D9.2)
 2030 FORMAT(1X,24HTEST FOR MULTIPLICITIES.,10X,D9.2)
 2040 FORMAT(1X,23HRADIUS TO START SEARCH.,11X,D9.2)
 2050 FORMAT(1X,21HRADIUS TO END SEARCH.,13X,D9.2)
 2060 FORMAT(//1X)
 2070 FORMAT(1X,31HTEST FOR ZERO IN SUBROUTINE GCD,3X,D9.2)
      END
```

TABLE XVIII-B (Continued)

```
            SUBROUTINE METHOD(RA,VA,N,RXZERO,VXZERO,NIAP,RX,VX,K,RXINIT,VXINIT
           1,RD,VD,MULT,ND,IROOT,EPSQ,XSTART,XEND,EPSCNV,EPSMUL)
      C    ***********************************************************************
      C    *                                                                     *
      C    * THIS SUBROUTINE USES NEWTON'S METHOD TO EXTRACT THE ZEROS OF A      *
      C    * POLYNOMIAL. A SEQUENCE OF APPROXIMATIONS IS CONSTRUCTED CONVERGING TO A  *
      C    * ZERO OF THE POLYNOMIAL BY USING THE ITERATION FORMULA               *
      C    *               X(N+1) = X(N)-P(X(N))/P'(X(N)).                       *
      C    *                                                                     *
      C    ***********************************************************************
            DOUBLE PRECISION RA,VA,RXZERO,VXZERO,RB,VB,RCOEF,VCOEF,RX,VX,RXINI
           1T,VXINIT,RC,VC,RD,VD,RPX,VPX,RDPX,VDPX,RXNEW,VXNEW,RXO,VXO,EPSCHK,
           2EPSCNV,EPSQ,EPSMUL,XSTART,XEND,ABPX,ABDPX
            DOUBLE PRECISION DSQRT
            DIMENSION RA(26),VA(26),RB(26),VB(26),RC(26),VC(26),RD(26),VD(26),
           1RCOEF(26),VCOEF(26),MULT(25),RXZERO(25),VXZERO(25),RX(25),VX(25),R
           2XINIT(25),VXINIT(25)
            COMMON EPSCHK,IO2,MAX
            IF(NIAP.NE.0) GO TO 1
            NIAP=N
            CALL GENAPP(RXZERO,VXZERO,NIAP,XSTART)
          1 NA=N+1                                                                130
            NDEF=N
            L=1                                                                   152
            ITER=0                                                                154
            NROOT=0                                                               160
            IROOT=0
            ITIME=0
            ND=0
            IALTER=0                                                              164
            K=0                                                                   168
            RXO=RXZERO(L)                                                         180
            VXO=VXZERO(L)                                                         181
            DO 5 I=1,NA                                                           188
            RCOEF(I)=RA(I)                                                        190
          5 VCOEF(I)=VA(I)                                                        191
         10 CALL HORNER(RCOEF,VCOEF,RXO,VXO,NDEF,RB,VB,RC,VC,RPX,VPX,RDPX,VDPX
           1)
            ABPX=DSQRT(RPX*RPX+VPX*VPX)
            ABDPX=DSQRT(RDPX*RDPX+VDPX*VDPX)
            IF(ABDPX.NE.0.0) GO TO 20
            IF(ABPX.EQ.0.0) GO TO 70
            GO TO 110
         20 CALL NEWTON(RPX,VPX,RDPX,VDPX,RXO,VXO,RXNEW,VXNEW)
            ITER=ITER+1                                                           200
            RXO=RXNEW                                                             204
            VXO=VXNEW                                                             205
            EPSCHK=EPSCNV
            CALL CHECK(RPX,VPX,RDPX,VDPX,RXO,VXO,KANS)
            IF(KANS.EQ.1) GO TO 70                                                194
            IF(ITER.GE.MAX) GO TO 40
            GO TO 10                                                              208
         40 CALL ALTER(RXZERO(L),VXZERO(L),IALTER,ITIME)
            IF(IALTER.GT.5) GO TO 110
            RXO=RXZERO(L)
            VXO=VXZERO(L)
            ITER=0                                                                244
            GO TO 10                                                              248
         60 ND=NDEF+1
            DO 65 J=1,ND
```

TABLE XVIII-B (Continued)

```
      RD(J)=RCOEF(J)
   65 VD(J)=VCOEF(J)
      GO TO 140
   70 NROOT=NROOT+1                                                      268
      K=K+1                                                              272
      MULT(K)=1                                                          276
      RX(K)=RXO                                                          280
      VX(K)=VXO                                                          281
      RXINIT(K)=RXZERO(L)                                                288
      VXINIT(K)=VXZERO(L)                                                289
      CALL HORNER(RCOEF,VCOEF,RXO,VXO,NDEF,RB,VB,RC,VC,RPX,VPX,RDPX,VDPX
     1)
   80 IF(NROOT.GE.N) GO TO 147
      NDEF=NDEF-1
      NUM=NDEF+1
      DO 105 I=1,NUM                                                     294
      RCOEF(I)=RB(I)                                                     296
  105 VCOEF(I)=VB(I)                                                     297
      CALL HORNER(RCOEF,VCOEF,RXO,VXO,NDEF,RB,VB,RC,VC,RPX,VPX,RDPX,VDPX
     1)
      ABPX=DSQRT(RPX*RPX+VPX*VPX)
      ABDPX=DSQRT(RDPX*RDPX+VDPX*VDPX)
      IF(ABDPX.NE.0.0) GO TO 107
      IF(ABPX.EQ.0.0) GO TO 130
      GO TO 110
  107 CONTINUE
      EPSCHK=EPSMUL
      CALL CHECK(RPX,VPX,RDPX,VDPX,RXO,VXO,KANS)
      IF(KANS.EQ.1) GO TO 130                                            300
  110 IF(NDEF.GT.2) GO TO 113
      IROOT=K
      CALL QUAD(RCOEF,VCOEF,NDEF,RX,VX,K,MULT,EPSQ)
      GO TO 150
  113 IF(L.LT.NIAP) GO TO 115
      IF(XEND.EQ.0.0) GO TO 60
      IF(XSTART.GT.XEND) GO TO 60
      NIAP=N
      CALL GENAPP(RXZERO,VXZERO,NIAP,XSTART)
      L=0
  115 L=L+1
      RXO=RXZERO(L)                                                      312
      VXO=VXZERO(L)                                                      313
      ITER=0                                                             316
      IALTER=0                                                           320
      GO TO 10                                                           324
  130 MULT(K)=MULT(K)+1                                                  328
      NROOT=NROOT+1                                                      332
      GO TO 80                                                           336
  140 IF(K.EQ.0) GO TO 160                                               338
  147 IROOT=K
  150 WRITE(IO2,1025)
      WRITE(IO2,1050)
      WRITE(IO2,1060) (I,RX(I),VX(I),MULT(I),RXINIT(I),VXINIT(I),I=1,IRO
     1OT)
      KKK=IROOT+1
      IF(IROOT.LT.K) WRITE(IO2,1062) (I,RX(I),VX(I),MULT(I),I=KKK,K)
      EPSCHK=EPSCNV
      CALL BETTER(K,RXZERO,VXZERO,RX,VX,NA,RA,VA,RCOEF,VCOEF,N,RC,VC,RB,
     1VB)
  160 RETURN
```

TABLE XVIII-B (Continued)

```
      1025 FORMAT(///1X,61HBEFORE THE ATTEMPT TO IMPROVE ACCURACY, THE ZEROS
          1OF P(X) ARE)
      1050 FORMAT(///2X,13HROOTS OF P(X),52X,14HMULTIPLICITIES,17X,21HINITIAL
          1 APPROXIMATION//)
      1060 FORMAT(3X,5HROOT(,I2,4H) = ,D23.16,3H + ,D23.16,2H I,7X,I2,7X,D23.
          116,3H + ,D23.16,2H I)
      1062 FORMAT(3X,5HROOT(,I2,4H) = ,D23.16,3H + ,D23.16,2H I,7X,I2,8X,23HS
          1OLVED BY DIRECT METHOD)
           END                                                               450
```

```
           SUBROUTINE GENAPP(APPR,APPI,NAPP,XSTART)
C     **********************************************************************
C     *                                                                    *
C     * SUBROUTINE GENAPP GENERATES N INITIAL APPROXIMATIONS, WHERE N IS THE *
C     * DEGREE OF THE ORIGINAL POLYNOMIAL.                                 *
C     *                                                                    *
C     **********************************************************************
           DOUBLE PRECISION APPR,APPI,XSTART,DUMMY,BETA
           DOUBLE PRECISION DCOS,DSIN
           DIMENSION APPR(25),APPI(25)
           COMMON DUMMY,IO2,MAX
           IF(XSTART.EQ.0.0) XSTART=0.5
           BETA=0.2617994
           DO 10 I=1,NAPP
           APPR(I)=XSTART*DCOS(BETA)
           APPI(I)=XSTART*DSIN(BETA)
           BETA=BETA+0.5235988
        10 XSTART=XSTART+0.5
           RETURN
           END
```

TABLE XVIII-B (Continued)

```
      SUBROUTINE ALTER(XOLDR,XOLDI,NALTER,ITIME)
C    ************************************************************************
C    *                                                                      *
C    * SUBROUTINE ALTER ALTERS THE INITIAL APPROXIMATIONS WHICH PRODUCE NO   *
C    * CONVERGENCE TO A ZERO. THIS IS DONE A MAXIMUM OF 5 TIMES FOR EACH ROOT. *
C    *                                                                      *
C    ************************************************************************
      DOUBLE PRECISION XOLDR,XOLDI,DUMMY,ABXOLD,BETA
      DOUBLE PRECISION DCOS,DSIN
      DOUBLE PRECISION DATAN2
      DOUBLE PRECISION DSQRT
      COMMON DUMMY,IO2,MAX
      IF(ITIME.NE.0) GO TO 5
      ITIME =1
      WRITE(IO2,1010) MAX
    5 IF(NALTER.EQ.0) GO TO 10
      WRITE(IO2,1000) XOLDR,XOLDI
      GO TO 20
   10 ABXOLD=DSQRT(XOLDR*XOLDR+XOLDI*XOLDI)
      BETA=DATAN2(XOLDI,XOLDR)
      WRITE(IO2,1020) XOLDR,XOLDI
   20 NALTER=NALTER+1
      IF(NALTER.GT.5) RETURN
      GO TO (30,40,30,40,30),NALTER
   30 XOLDR=-XOLDR
      XOLDI=-XOLDI
      GO TO 50
   40 BETA=BETA+1.0471976
      XOLDR=ABXOLD*DCOS(BETA)
      XOLDI=ABXOLD*DSIN(BETA)
   50 RETURN
 1000 FORMAT(1X,D23.16,3H + ,D23.16,2H I,10X,21HALTERED APPROXIMATION)
 1010 FORMAT(///1X,54HNO CONVERGENCE FOR THE FOLLOWING APPROXIMATIONS AF
     1TER ,I3,12H ITERATIONS.//)
 1020 FORMAT(/1X,D23.16,3H + ,D23.16,2H I,10X,21HINITIAL APPROXIMATION)
      END
```

TABLE XVIII-B (Continued)

```
      SUBROUTINE QUAD(UA,VA,NA,UROOT,VROOT,NROOT,MULTI,EPST)
C     ***********************************************************************
C     *                                                                     *
C     * SUBROUTINE QUAD SOLVES DIRECTLY FOR THE ZEROS AND THEIR MULTIPLICITIES *
C     * OF EITHER A QUADRATIC POLYNOMIAL OR A LINEAR FACTOR.  SOLUTION OF THE  *
C     * QUADRATIC IS DONE USING THE QUADRATIC FORMULA.                        *
C     *                                                                     *
C     ***********************************************************************
      DOUBLE PRECISION UA,VA,UROOT,VROOT,BBB,UAAA,VAAA,UDISC,VDISC,UDUMM
     1Y,VDUMMY,RDUMMY,SDUMMY,EPST,UBBB,VBBB
      DOUBLE PRECISION DSQRT
      DIMENSION UA(26),VA(26),UROOT(25),VROOT(25),MULTI(25)
      IF(NA.EQ.2) GO TO 7
      IF(NA.EQ.1) GO TO 5
      UROOT(NROOT+1)=0.0
      VROOT(NROOT+1)=0.0
      MULTI(NROOT+1)=1
      NROOT=NROOT+1
      GO TO 50
    5 BBB=UA(1)*UA(1)+VA(1)*VA(1)
      UROOT(NROOT+1)=(-UA(2)*UA(1)-VA(2)*VA(1))/BBB
      VROOT(NROOT+1)=(-VA(2)*UA(1)+UA(2)*VA(1))/BBB
      MULTI(NROOT+1)=1
      NROOT=NROOT+1
      GO TO 50
    7 UDISC=(UA(2)*UA(2)-VA(2)*VA(2))-(4.0*(UA(1)*UA(3)-VA(1)*VA(3)))
      VDISC=(VA(2)*UA(2)+UA(2)*VA(2))-(4.0*(VA(1)*UA(3)+UA(1)*VA(3)))
      BBB=DSQRT(UDISC*UDISC+VDISC*VDISC)
      IF(BBB.LT.EPST) GO TO 10
      CALL COMSQT(UDISC,VDISC,UDUMMY,VDUMMY)
      UBBB=-UA(2)+UDUMMY
      VBBB=-VA(2)+VDUMMY
      RDUMMY=-UA(2)-UDUMMY
      SDUMMY=-VA(2)-VDUMMY
      UAAA=2.0*UA(1)
      VAAA=2.0*VA(1)
      BBB=UAAA*UAAA+VAAA*VAAA
      UROOT(NROOT+1)=(UBBB*UAAA+VBBB*VAAA)/BBB
      VROOT(NROOT+1)=(VBBB*UAAA-UBBB*VAAA)/BBB
      UROOT(NROOT+2)=(RDUMMY*UAAA+SDUMMY*VAAA)/BBB
      VROOT(NROOT+2)=(SDUMMY*UAAA-RDUMMY*VAAA)/BBB
      MULTI(NROOT+1)=1
      MULTI(NROOT+2)=1
      NROOT=NROOT+2
      GO TO 50
   10 UAAA=2.0*UA(1)
      VAAA=2.0*VA(1)
      BBB=UAAA*UAAA+VAAA*VAAA
      UROOT(NROOT+1)=(-UA(2)*UAAA-VA(2)*VAAA)/BBB
      VROOT(NROOT+1)=(-VA(2)*UAAA+UA(2)*VAAA)/BBB,
      MULTI(NROOT+1)=2
      NROOT=NROOT+1
   50 RETURN
      END
```

TABLE XVIII-B (Continued)

```
      SUBROUTINE HORNER(RA,VA,RXO,VXO,NDEF,RB,VB,RC,VC,RPX,VPX,RDPX,VDPX
     1)
C     ************************************************************************
C     *                                                                      *
C     * HORNER'S METHOD COMPUTES THE VALUE OF A POLYNOMIAL P(X) AT A POINT D AND *
C     * ITS DERIVATIVE AT D.  SYNTHETIC DIVISION IS USED TO DEFLATE THE      *
C     * POLYNOMIAL BY DIVIDING OUT THE FACTOR (X-D).                         *
C     *                                                                      *
C     ************************************************************************
      DOUBLE PRECISION VDPX,RXO,VXO,RB,VB,RC,VC,RPX,VPX,RDPX,RA,VA
      DIMENSION RA(26),VA(26),RB(26),VB(26),RC(26),VC(26)
      RB(1)=RA(1)                                                        516
      VB(1)=VA(1)                                                        517
      NUM=NDEF+1                                                         520
      DO 10 I=2,NUM                                                      524
      RB(I)=RA(I)+(RB(I-1)*RXO-VB(I-1)*VXO)
   10 VB(I)=VA(I)+(VB(I-1)*RXO+RB(I-1)*VXO)
      RPX=RB(NUM)                                                        532
      VPX=VB(NUM)                                                        533
      RC(1)=RB(1)                                                        540
      VC(1)=VB(1)                                                        541
      IF(NDEF.LT.2) GO TO 25
      DO 20 J=2,NDEF                                                     544
      RC(J)=RB(J)+(RC(J-1)*RXO-VC(J-1)*VXO)
   20 VC(J)=VB(J)+(VC(J-1)*RXO+RC(J-1)*VXO)
   25 RDPX=RC(NDEF)                                                      552
      VDPX=VC(NDEF)                                                      553
      RETURN                                                             572
      END                                                                580
```

```
      SUBROUTINE NEWTON(RPX,VPX,RDPX,VDPX,RXO,VXO,RXNEW,VXNEW)           600
C     ************************************************************************
C     *                                                                      *
C     * THIS SUBROUTINE CALCULATES A NEW APPROXIMATION FROM THE OLD APPROX-  *
C     * IMATION BY USING THE ITERATION FORMULA                              *
C     *               X(N+1) = X(N)-P(X(N))/P'(X(N)).                        *
C     *                                                                      *
C     ************************************************************************
      DOUBLE PRECISION RPX,VPX,RDPX,VDPX,RXO,VXO,RXNEW,VXNEW,ARG
      DOUBLE PRECISION DSQRT
      DOUBLE PRECISION DDD
      ARG=RDPX*RDPX+VDPX*VDPX
      DDD=DSQRT(ARG)
      IF(DDD.EQ.0.0) RETURN
      RXNEW=RXO-((RPX*RDPX+VPX*VDPX)/ARG)
      VXNEW=VXO-((VPX*RDPX-RPX*VDPX)/ARG)
      RETURN                                                             616
      END                                                                620
```

TABLE XVIII-B (Continued)

```
      SUBROUTINE CHECK(RPX,VPX,RDPX,VDPX,RXO,VXO,KANS)
C   ***********************************************************************
C   *                                                                     *
C   * THIS SUBROUTINE CHECKS FOR CONVERGENCE OF THE SEQUENCE OF APPROX-   *
C   * IMATIONS BY TESTING THE EXPRESSION                                  *
C   * ABSOLUTE VALUE OF (P(X(N))/P'(X(N)))/ABSOLUTE VALUE OF X(N+1).      *
C   * WHEN IT IS AS SMALL AS DESIRED, CONVERGENCE IS OBTAINED.            *
C   *                                                                     *
C   ***********************************************************************
      DOUBLE PRECISION RPX,VPX,RDPX,VDPX,RXO,VXO,ABSXO,ABSQUO,RDUMMY,VDU     749
     1MMY,EPS                                                                750
      DOUBLE PRECISION DDD
      DOUBLE PRECISION ARG
      DOUBLE PRECISION DSQRT
      COMMON EPS,IO2,MAX                                                     751
      ABSXO=DSQRT(RXO*RXO+VXO*VXO)
      IF(ABSXO.EQ.0.) GO TO 25
      ARG=RDPX*RDPX+VDPX*VDPX
      DDD=DSQRT(ARG)
      IF(DDD.EQ.0.0) GO TO 25
      RDUMMY=(RPX*RDPX+VPX*VDPX)/ARG
      VDUMMY=(VPX*RDPX-RPX*VDPX)/ARG
      ABSQUO=DSQRT(RDUMMY*RDUMMY+VDUMMY*VDUMMY)
      IF(ABSQUO/ABSXO.LT.EPS) GO TO 10
      KANS=0                                                                 760
      RETURN                                                                 764
   10 KANS=1                                                                 768
      RETURN                                                                 772
   25 KANS=0
      RETURN
      END                                                                    780
```

TABLE XVIII-B (Continued)

```
      SUBROUTINE BETTER(K,RXZERO,VXZERO,RX,VX,NA,RA,VA,RCOEF,VCOEF,N,RC,      800
     1VC,RB,VB)
C     ************************************************************************
C     *                                                                      *
C     * SUBROUTINE BETTER ATTEMPTS TO IMPROVE THE ACCURACY OF THE ZEROS FOUND  *
C     * BY USING THEM AS INITIAL APPROXIMATIONS WITH NEWTON'S METHOD APPLIED TO *
C     * THE FULL, UNDEFLATED POLYNOMIAL.                                     *
C     *                                                                      *
C     ************************************************************************
      DOUBLE PRECISION RXZERO,VXZERO,RX,VX,RA,VA,RCOEF,VCOEF,RC,VC,RB,VB      805
     1,RXO,VXO,RPX,VPX,RDPX,VDPX,RXNEW,VXNEW,EPS                              806
      DOUBLE PRECISION ABPX,ABDPX
      DOUBLE PRECISION DSQRT
      DIMENSION RXZERO(25),VXZERO(25),RX(25),VX(25),RA(26),VA(26),RCOEF(      808
     126),VCOEF(26),RC(26),VC(26),RB(26),VB(26)
      COMMON EPS,IO2,MAX
      DO 10 I=1,K                                                             812
      RXZERO(I)=RX(I)                                                         815
   10 VXZERO(I)=VX(I)                                                         816
      DO 20 I=1,NA
      RCOEF(I)=RA(I)                                                          824
   20 VCOEF(I)=VA(I)                                                          825
      DO 50 J=1,K                                                             828
      RXO=RXZERO(J)                                                           832
      VXO=VXZERO(J)                                                           833
      NN=N                                                                    834
      ITER=0                                                                  836
   30 CALL HORNER(RCOEF,VCOEF,RXO,VXO,NN,RB,VB,RC,VC,RPX,VPX,RDPX,VDPX)
      ABPX=DSQRT(RPX*RPX+VPX*VPX)
      ABDPX=DSQRT(RDPX*RDPX+VDPX*VDPX)
      IF(ABDPX.NE.0.0) GO TO 33
      IF(ABPX.EQ.0.0) GO TO 40
      GO TO 34
   33 CALL NEWTON(RPX,VPX,RDPX,VDPX,RXO,VXO,RXNEW,VXNEW)
      ITER=ITER+1                                                             856
      RXO=RXNEW                                                               860
      VXO=VXNEW                                                               861
      CALL CHECK(RPX,VPX,RDPX,VDPX,RXO,VXO,KANS)
      IF(KANS.EQ.1) GO TO 40                                                  844
      IF(ITER.GE.MAX) GO TO 35
      GO TO 30                                                                864
   34 WRITE(IO2,1112) RXO,VXO
   35 WRITE(IO2,100) J,RXZERO(J),VXZERO(J)
      WRITE(IO2,200) MAX
   40 RX(J)=RXO                                                               870
      VX(J)=VXO                                                               871
   50 CONTINUE                                                                872
      RETURN                                                                  876
 1112 FORMAT(1H0,36HTHE VALUE OF THE DERIVATIVE AT XO = ,D23.16,3H + ,D2
     13.16,2H I,10H  IS ZERO.)
  100 FORMAT(42H0IN THE ATTEMPT TO IMPROVE ACCURACY, ROOT(,I2,4H) = ,D23
     1.16,3H + ,D23.16,2H I,18H DID NOT CONVERGE.)
  200 FORMAT(33H THE PRESENT APPROXIMATION AFTER ,I3,29H ITERATIONS IS P
     1RINTED BELOW.)
      END                                                                     880
```

TABLE XVIII-B (Continued)

```
      SUBROUTINE COMSQT(UX,VX,UY,VY)
C    ***************************************************************************
C    *                                                                         *
C    * THIS SUBROUTINE COMPUTES THE SQUARE ROOT OF A COMPLEX NUMBER.           *
C    *                                                                         *
C    ***************************************************************************
      DOUBLE PRECISION UX,VX,UY,VY,DUMMY,R,AAA,BBB
      DOUBLE PRECISION DSQRT,DABS
      R=DSQRT(UX*UX+VX*VX)
      AAA=DSQRT(DABS((R+UX)/2.0))
      BBB=DSQRT(DABS((R-UX)/2.0))
      IF(VX) 10,20,30
   10 UY=AAA
      VY=-1.0*BBB
      GO TO 100
   20 IF(UX) 40,50,60
   30 UY=AAA
      VY=BBB
      GO TO 100
   40 DUMMY=DABS(UX)
      UY=0.0
      VY=DSQRT(DUMMY)
      GO TO 100
   50 UY=0.0
      VY=0.0
      GO TO 100
   60 DUMMY=DABS(UX)
      UY=DSQRT(DUMMY)
      VY=0.0
  100 RETURN
      END
```

```
      SUBROUTINE DERIV(UA,VA,NA,UB,VB,NB)
C    ***************************************************************************
C    *                                                                         *
C    * GIVEN A POLYNOMIAL P(X), SUBROUTINE DERIV COMPUTES THE COEFFICIENTS OF  *
C    * ITS DERIVATIVE P'(X).                                                   *
C    *                                                                         *
C    ***************************************************************************
      DOUBLE PRECISION UA,VA,UB,VB
      DIMENSION UA(26),VA(26),UB(26),VB(26)
      IF(NA.GE.2) GO TO 30
      IF(NA.EQ.1) GO TO 20
      UB(1)=0.0
      VB(1)=0.0
      NB=-1
      GO TO 50
   20 UB(1)=UA(1)
      VB(1)=VA(1)
      NB=0
      GO TO 50
   30 NB=NA-1
      DO 40 I=1,NA
      BBB=NA-I+1
      UB(I)=BBB*UA(I)
   40 VB(I)=BBB*VA(I)
   50 RETURN
      END
```

TABLE XVIII-B (Continued)

```
      SUBROUTINE DIVIDE(UF,VF,NF,UG,VG,NG,UH,VH,NH)
C     ************************************************************************
C     *                                                                      *
C     * GIVEN TWO POLYNOMIALS F(X) AND G(X), SUBROUTINE DIVIDE COMPUTES THE   *
C     * QUOTIENT POLYNOMIAL  H(X) = F(X)/G(X).                                *
C     *                                                                      *
C     ************************************************************************
      DOUBLE PRECISION UF,VF,UG,VG,UH,VH,UX,VX,UC,VC,UPX,VPX,UDPX,VDPX,U
     1TERM,VTERM,UDUMMY,VDUMMY,EPS
      DOUBLE PRECISION DENOM
      DIMENSION UF(26),VF(26),UG(26),VG(26),UH(26),VH(26),UC(26),VC(26)
      COMMON EPS,IO2,MAX
      IF(NG.GE.2) GO TO 60
      IF(NG.EQ.1) GO TO 30
      NH=NF
      KKK=NF+1
      DO 20 I=1,KKK
      DENOM=UG(1)*UG(1)+VG(1)*VG(1)
      UH(I)=(UF(I)*UG(1)+VF(I)*VG(1))/DENOM
   20 VH(I)=(VF(I)*UG(1)-UF(I)*VG(1))/DENOM
      GO TO 100
   30 UDUMMY=-1.0*UG(2)
      VDUMMY=-1.0*VG(2)
      DENOM=UG(1)*UG(1)+VG(1)*VG(1)
      UX=(UDUMMY*UG(1)+VDUMMY*VG(1))/DENOM
      VX=(VDUMMY*UG(1)-UDUMMY*VG(1))/DENOM
      NNF=NF
      CALL HORNER(UF,VF,UX,VX,NNF,UH,VH,UC,VC,UPX,VPX,UDPX,VDPX)
      NH=NNF-1
      GO TO 100
   60 NH=NF-NG
      DENOM=UG(1)*UG(1)+VG(1)*VG(1)
      UH(1)=(UF(1)*UG(1)+VF(1)*VG(1))/DENOM
      VH(1)=(VF(1)*UG(1)-UF(1)*VG(1))/DENOM
      KKK=NH+1
      DO 95 I=2,KKK
      UTERM=UF(I)
      VTERM=VF(I)
      J=2
      K=I-1
   70 UTERM=UTERM-(UG(J)*UH(K)-VG(J)*VH(K))
      VTERM=VTERM-(VG(J)*UH(K)+UG(J)*VH(K))
      IF(J.EQ.I) GO TO 90
      IF(J.GE.NG+1) GO TO 90
      J=J+1
      K=K-1
      GO TO 70
   90 DENOM=UG(1)*UG(1)+VG(1)*VG(1)
      UH(I)=(UTERM*UG(1)+VTERM*VG(1))/DENOM
   95 VH(I)=(VTERM*UG(1)-UTERM*VG(1))/DENOM
  100 RETURN
      END
```

TABLE XVIII-B (Continued)

```
      SUBROUTINE GCD(UP,VP,NP,UDP,VDP,NDP,URK,VRK,NK,EPSG)
C     ***********************************************************************
C     *                                                                     *
C     * GIVEN POLYNOMIALS P(X) AND DP(X) WHERE DEG. DP(X) IS LESS THAN DEG. *
C     * P(X), SUBROUTINE GCD COMPUTES THE GREATEST COMMON DIVISOR OF P(X) AND *
C     * DP(X).                                                              *
C     *                                                                     *
C     ***********************************************************************
      DOUBLE PRECISION EPSG,UP,VP,UDP,VDP,URK,VRK,UR1,VR1,UR2,VR2,UT,VT,
     1UU,VU,BBB,EPS
      DOUBLE PRECISION DENOM
      DOUBLE PRECISION DSQRT
      DIMENSION UR1(26),VR1(26),UR2(26),VR2(26),UT(26),VT(26),URK(26),VR
     1K(26),UP(26),VP(26),UDP(26),VDP(26)
      COMMON EPS,IO2,MAX
      N1=NP
      N2=NDP
      KKK=N1+1
      DO 10 I=1,KKK
      UR1(I)=UP(I)
   10 VR1(I)=VP(I)
      KKK=N2+1
      DO 20 I=1,KKK
      UR2(I)=UDP(I)
   20 VR2(I)=VDP(I)
   50 DENOM=UR2(1)*UR2(1)+VR2(1)*VR2(1)
      UU=(UR1(1)*UR2(1)+VR1(1)*VR2(1))/DENOM
      VU=(VR1(1)*UR2(1)-UR1(1)*VR2(1))/DENOM
      KKK=N2+1
      DO 70 I=1,KKK
      UT(I)=UR1(I)-(UU*UR2(I)-VU*VR2(I))
   70 VT(I)=VR1(I)-(VU*UR2(I)+UU*VR2(I))
      UT(1)=0.0
      VT(1)=0.0
      IF(N1.EQ.N2) GO TO 90
      KKK=N1+1
      NNN=N2+2
      DO 80 I=NNN,KKK
      UT(I)=UR1(I)
   80 VT(I)=VR1(I)
   90 KKK=N1+1
      DO 100 I=1,KKK
      BBB=DSQRT(UT(I)*UT(I)+VT(I)*VT(I))
      IF(BBB.GT.EPSG) GO TO 120
  100 CONTINUE
      NK=N2
      KKK=NK+1
      DO 110 I=2,KKK
      DENOM=UR2(1)*UR2(1)+VR2(1)*VR2(1)
      URK(I)=(UR2(I)*UR2(1)+VR2(I)*VR2(1))/DENOM
  110 VRK(I)=(VR2(I)*UR2(1)-UR2(I)*VR2(1))/DENOM
      URK(1)=1.0
      VRK(1)=0.0
      GO TO 200
  120 M=N1+1-I
      IF(M.LT.N2) GO TO 160
      N1=M
      KKK=N1+1
      NNN=I-1
      DO 150 J=1,KKK
```

TABLE XVIII-B (Continued)

```
      UR1(J)=UT(NNN+J)
  150 VR1(J)=VT(NNN+J)
      GO TO 50
  160 N1=N2
      KKK=N1+1
      DO 170 J=1,KKK
      UR1(J)=UR2(J)
  170 VR1(J)=VR2(J)
      N2=M
      KKK=N2+1
      NNN=I-1
      DO 180 J=1,KKK
      UR2(J)=UT(NNN+J)
  180 VR2(J)=VT(NNN+J)
      GO TO 50
  200 RETURN
      END


      SUBROUTINE MULTIP(UA,VA,NA,UROOT,VROOT,NR,MULTI,EPSPX)
C   ************************************************************************
C   *                                                                      *
C   * GIVEN NR ZEROS OF A POLYNOMIAL, SUBROUTINE MULTIP COMPUTES THEIR     *
C   * MULTIPLICITIES.                                                      *
C   *                                                                      *
C   ************************************************************************
      DOUBLE PRECISION UA,VA,UROOT,VROOT,EPSPX,UWORK,VWORK,UB,VB,UC,VC,U
     1PX,VPX,UDPX,VDPX,BBB,EPS
      DOUBLE PRECISION DSQRT
      DIMENSION UA(26),VA(26),UROOT(25),VROOT(25),UWORK(26),VWORK(26),MU
     1LTI(25),UB(26),VB(26),UC(26),VC(26)
      COMMON EPS,IO2,MAX
      DO 50 I=1,NR
      MULTI(I)=0
      KKK=NA+1
      DO 20 J=1,KKK
      UWORK(J)=UA(J)
   20 VWORK(J)=VA(J)
      NWORK=NA
   25 CALL HORNER(UWORK,VWORK,UROOT(I),VROOT(I),NWORK,UB,VB,UC,VC,UPX,VP
     1X,UDPX,VDPX)
      NWORK=NWORK-1
      BBB=DSQRT(UPX*UPX+VPX*VPX)
      IF(BBB.GT.EPSPX) GO TO 45
      MULTI(I)=MULTI(I)+1
      IF(NWORK.LT.1) GO TO 50
      KKK=NWORK+1
      DO 40 J=1,KKK
      UWORK(J)=UB(J)
   40 VWORK(J)=VB(J)
      GO TO 25
   45 IF(MULTI(I).EQ.0) WRITE(IO2,1000) EPSPX,I,UROOT(I),VROOT(I)
   50 CONTINUE
  100 RETURN
 1000 FORMAT(///14H THE EPSILON (,D10.03,49H) CHECK IN SUBROUTINE MULTIP
     1 INDICATES THAT ROOT(,I2,4H) = ,D23.16,3H + ,D23.16,2H I/80H IS NO
     2T CLOSE ENOUGH TO BE A TRUE ROOT.  IT IS PRINTED BELOW WITH MULTIP
     3LICITY 0//)
      END

$ENTRY
```

APPENDIX E

G.C.D. - MULLER'S METHOD

1. Use of the Programs

Two programs using the grestest common divisor method supported by Muller's method are presented here. The first is the single precision program. The second program is in double precision and is designed to perform double precision complex arithmetic. These programs are written for use on any computer using FORTRAN IV language. They have been tested on the IBM S/360 mod. 50 computer which has a 32 bit word. However, it may be necessary to change the system functions as described below. The single precision program may be changed to double precision as described below.

After selecting the desired program, the input data should be prepared as described in section 2.

Each program is designed to solve polynomials of degree 25 or less. Both the coefficient of the highest degree term and the constant coefficient should be non-zero. In order to solve polynomials of degree N where $N > 25$, certain array dimensions must be changed. These are listed in Table XIX.

TABLE XIX

PROGRAM CHANGES FOR SOLVING POLYNOMIALS OF DEGREE GREATER THAN 25 BY G.C.D. - MULLER'S METHOD

Single Precision	Double Precision
Main Program	
P(N+1)	UP(N+1),VP(N+1)
OIAPP(N)	UOIAPP(N),VOIAPP(N)
ROOT(N)	UROOT(N),VROOT(N)
MULTI(N)	MULTI(N)
DP(N+1)	UDP(N+1),VDP(N+1)
RK(N+1)	URK(N+1),VRK(N+1)
H(N+1)	UH(N+1),VH(N+1)
Subroutine MULTIP	
A(N+1)	UA(N+1),VA(N+1)
ROOT(N)	UROOT(N),VROOT(N)
WORK(N+1)	UWORK(N+1),VWORK(N+1)
MULTI(N)	MULTI(N)
B(N+1)	UB(N+1),VB(N+1)

Subroutines DERIV, DIVIDE, and GCD

See corresponding subroutines of Table XIV in Appendix D.

Subroutines MULLER, BETTER, GENAPP, HORNER and QUAD

See Main Program or corresponding subprograms of Table IX in Appendix C.

Certain computers may require that the system functions of Table X in Appendix C be changed in both the single precision and double precision programs.

When used on the IBM S/360 with the WATFOR compiler for FORTRAN IV, the system functions given in Tables X-A of Appendix C and XIV-A of Appendix D must be typed in a declaration statement. These also appear in the program listing. For use without the WATFOR compiler or on other computers, these system functions might have to be removed.

The single precision program may be converted to double precision for use on machines equipped to perform double precision complex arithmetic provided the following changes or their equivalent are made and the system functions of Table XX are used and typed in a declaration statement where necessary. The changes presented below are those required for the IBM S/360. A "c" denotes a complex number and an "r" denotes a real number. The format statements should be changed from E-type to D-type.

In the main program and each subprogram change COMPLEX $C_1, C_2, \ldots$ to COMPLEX*16 $C_1, C_2, \ldots$ and add IMPLICIT REAL*8(A-H,O-Z).

TABLE XX

SYSTEM FUNCTIONS FOR CONVERTING SINGLE PRECISION G.C.D. - MULLER'S METHOD TO DOUBLE PRECISION

Single Precision	changed to	Double Precision
	Subroutines GCD, MULTIP, MULLER, TEST, and QUAD	
CABS(c)	- absolute value -	CDABS(c)
	Subroutines CALC, GENAPP, ALTER, and QUAD	

See corresponding subprograms of Table XI in Appendix C.

2. Input Data for G.C.D. - Muller's Method

The input data for G.C.D. - Muller's method is prepared exactly as that for G.C.D. - Newton's method as described in Appendix D, § 2.

3. Variables Used in G.C.D. - Muller's Method

The definitions of the major variables used in the G.C.D. - Muller's Method are referenced in Table XXI. The symbols used in the referenced tables are described in Appendix B, § 3. For variables not listed see the corresponding subprogram of Table VIII in Appendix B.

TABLE XXI

VARIABLES USED IN G.C.D. - MULLER'S METHOD

Main Program and Subroutines DERIV,
DIVIDE, GCD, and MULTIP

See corresponding subprogram of Table XVIII in Appendix D.

Subroutines MULLER, BETTER, TEST,
ALTER, HORNER, and QUAD

See Main Program or corresponding subprogram of Table XIII in Appendix C.

4. Description of Program Output

The output from the G.C.D. - Muller's method is the same as that for G.C.D. - Newton's method as described in Appendix D, § 4.

5. Informative and Error Messages

The informative and error messages are the same as those for G.C.D. - Newton's method as described in Appendix D, § 5.

MAIN PROGRAM

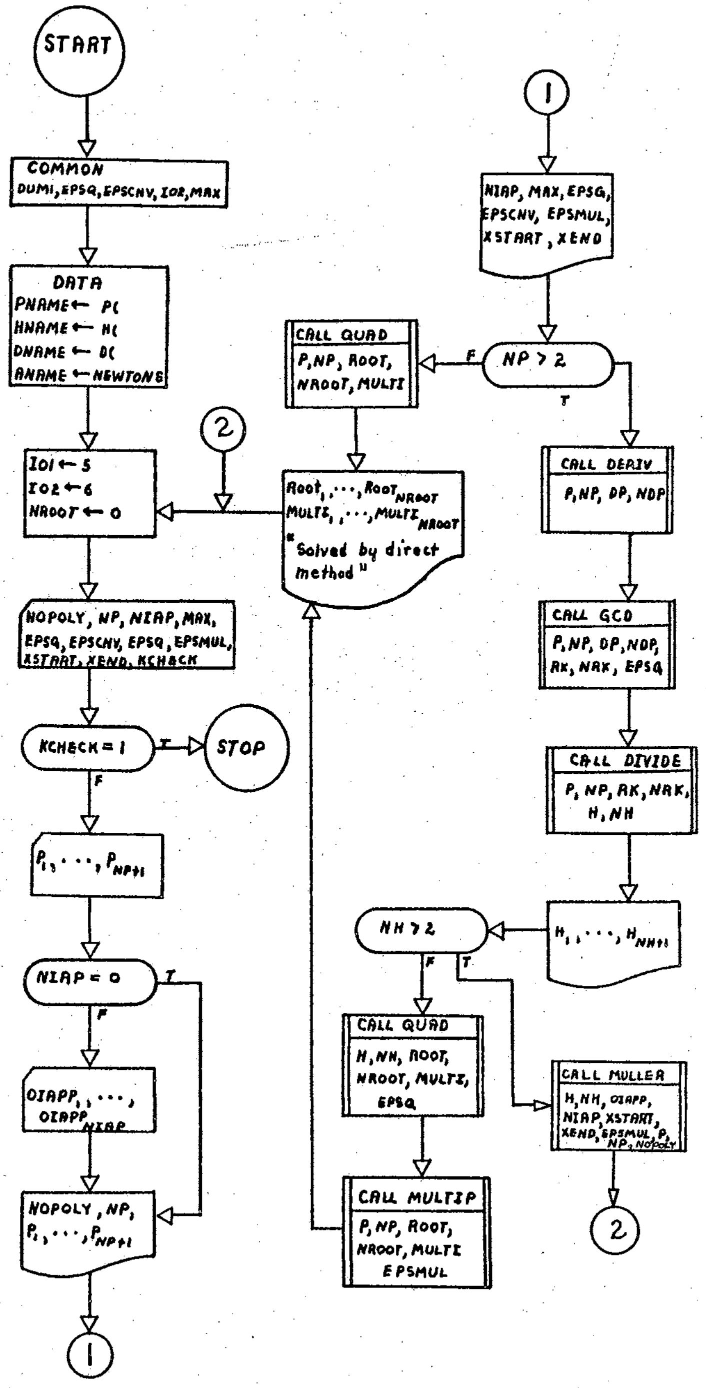
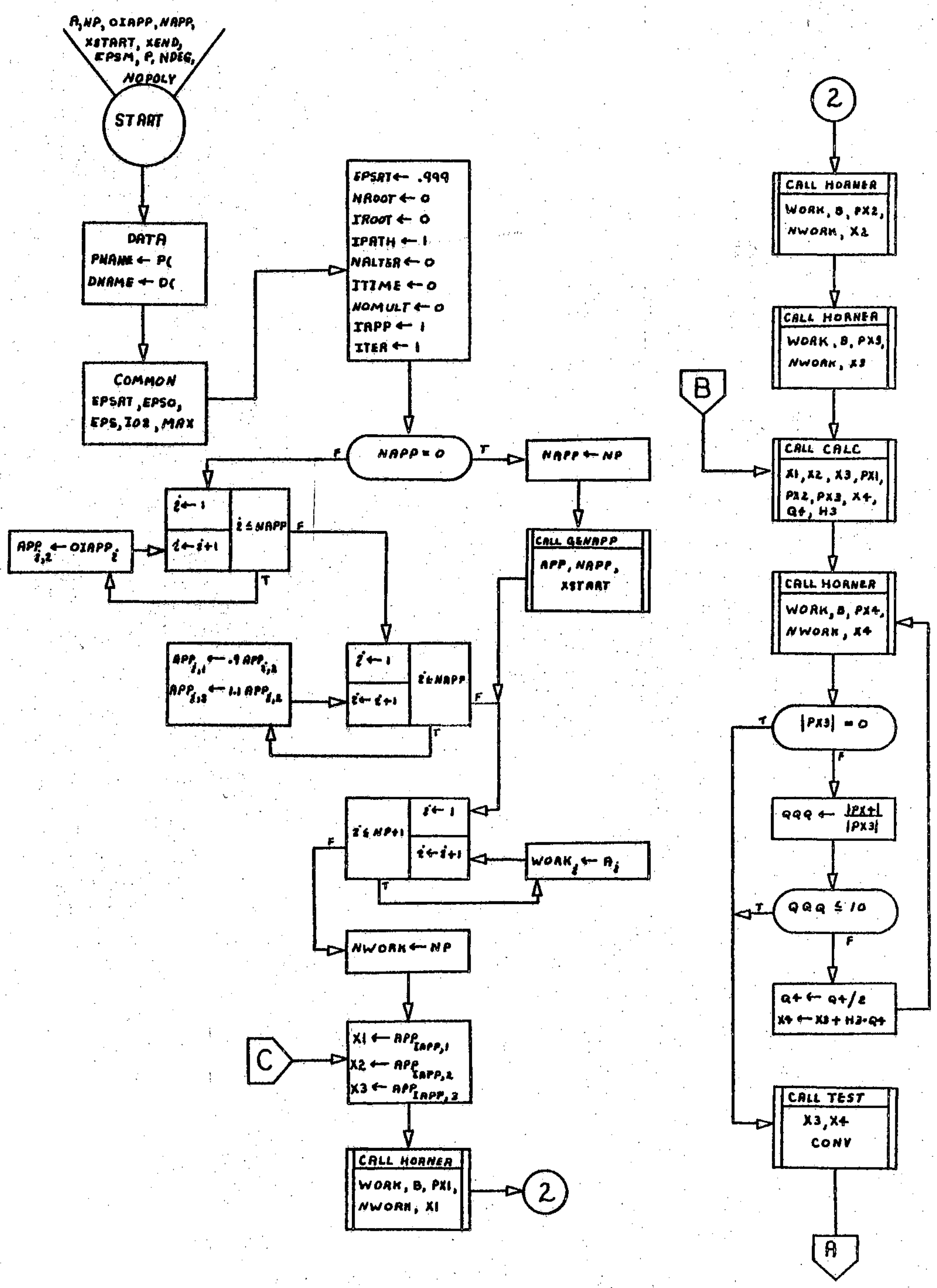

START
COMMON
DUMI, EPSQ, EPSCNV, IO2, MAX
DATA
PNAME ← P(
HNAME ← H(
DNAME ← D(
ANAME ← NEWTONS
IO1 ← 5
IO2 ← 6
NROOT ← 0
NOPOLY, NP, NIAP, MAX, EPSQ, EPSCNV, EPSQ, EPSMUL, XSTART, XEND, KCHECK
KCHECK = 1
T
STOP
F
P1, ..., PNP+1
NIAP = 0
T
F
OIAPP1, ..., OIAPPNIAP
NOPOLY, NP, P1, ..., PNP+1
1
2
CALL QUAD
P, NP, ROOT, NROOT, MULTI
Root1, ..., RootNROOT
MULTI1, ..., MULTINROOT
"Solved by direct method"
1
NIAP, MAX, EPSQ, EPSCNV, EPSMUL, XSTART, XEND
NP > 2
F
T
CALL DERIV
P, NP, DP, NDP
CALL GCD
P, NP, DP, NDP, RK, NRK, EPSQ
CALL DIVIDE
P, NP, RK, NRK, H, NH
H1, ..., HNH+1
NH > 2
F
T
CALL QUAD
H, NH, ROOT, NROOT, MULTI, EPSQ
CALL MULTIP
P, NP, ROOT, NROOT, MULTI EPSMUL
CALL MULLER
H, NH, OIAPP, NIAP, XSTART, XEND, EPSMUL, P, NP, NOPOLY
2

Figure 13.2. Flow Charts for G.C.D. - Muller's Method

MULLER



A, NP, OIAPP, NAPP, XSTART, XEND, EPSM, P, NDEG, NOPOLY
START
DATA PNAME ← P(DNAME ← D(
COMMON EPSRT, EPSO, EPS, IO1, MAX
EPSRT ← .999
NROOT ← 0
IROOT ← 0
IPATH ← 1
NALTER ← 0
ITIME ← 0
NOMULT ← 0
IAPP ← 1
ITER ← 1
NAPP = 0
NAPP ← NP
CALL GENAPP APP, NAPP, XSTART
i ← 1
i ← i+1
i ≤ NAPP
APP_{i,2} ← OIAPP_i
APP_{i,1} ← .9 APP_{i,2}
APP_{i,3} ← 1.1 APP_{i,2}
i ≤ NP+1
WORK_i ← A_i
NWORK ← NP
X1 ← APP_{IAPP,1}
X2 ← APP_{IAPP,2}
X3 ← APP_{IAPP,3}
CALL HORNER WORK, B, PX1, NWORK, X1
CALL HORNER WORK, B, PX2, NWORK, X2
CALL HORNER WORK, B, PX3, NWORK, X3
CALL CALC X1, X2, X3, PX1, PX2, PX3, X4, Q4, H3
CALL HORNER WORK, B, PX4, NWORK, X4
|PX3| = 0
QQQ ← |PX4| / |PX3|
QQQ ≤ 10
Q4 ← Q4/2
X4 ← X3 + H3·Q4
CALL TEST X3, X4, CONV
T
F
2
A
B
C

Figure 13.2. (Continued)

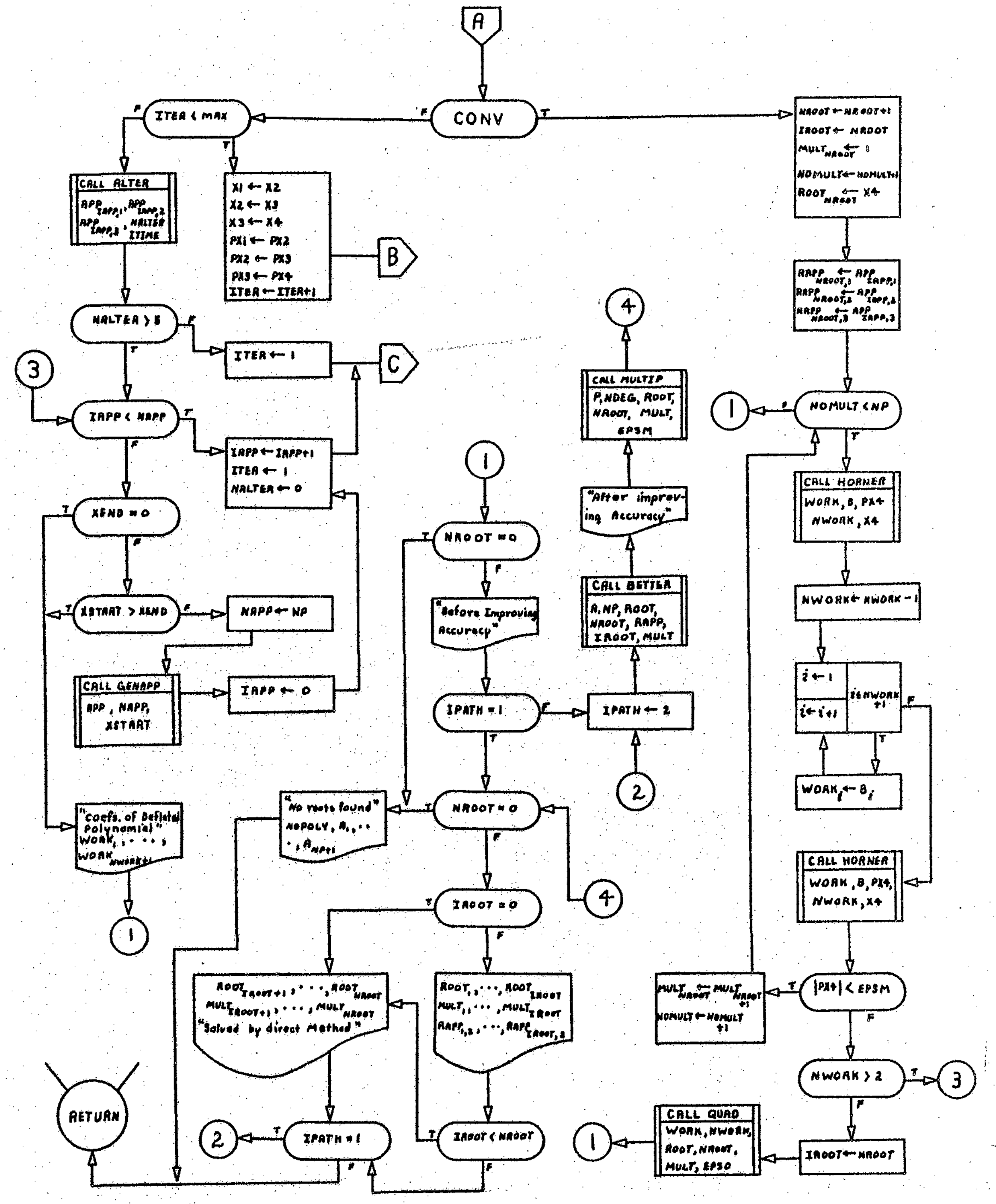


Figure 13.2. (Continued)

GCD

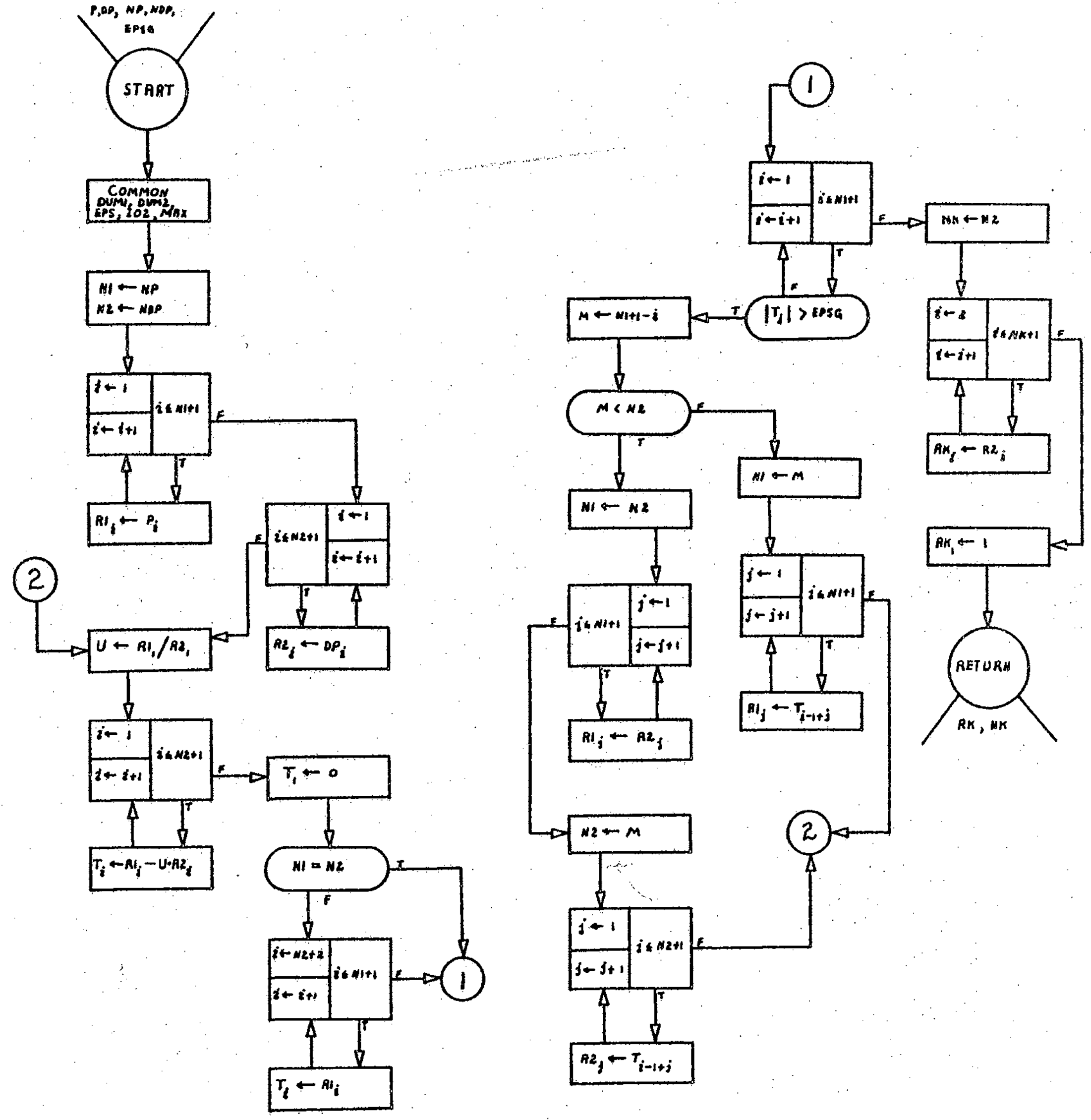


Figure 13.2. (Continued)

BETTER HORNER

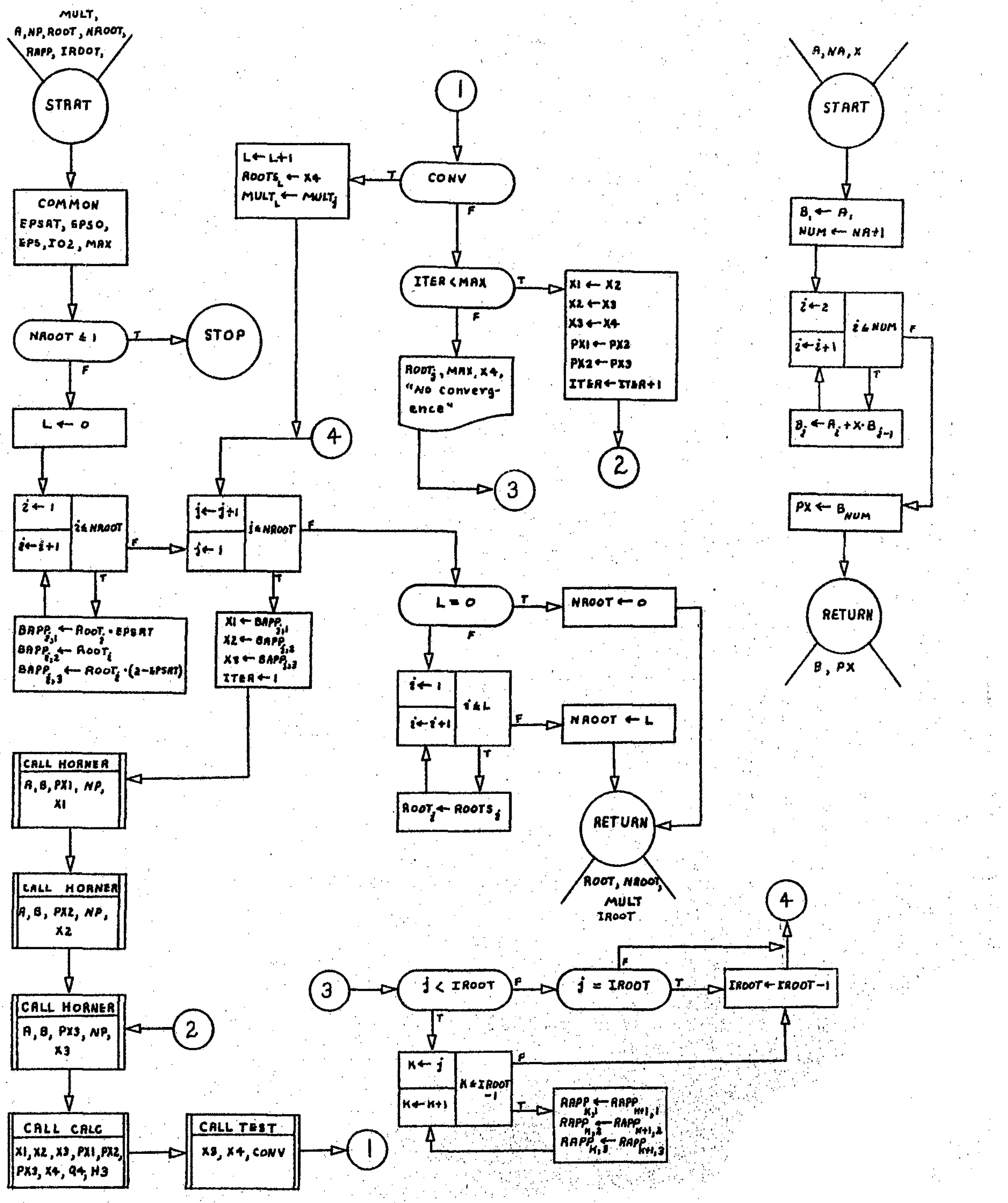
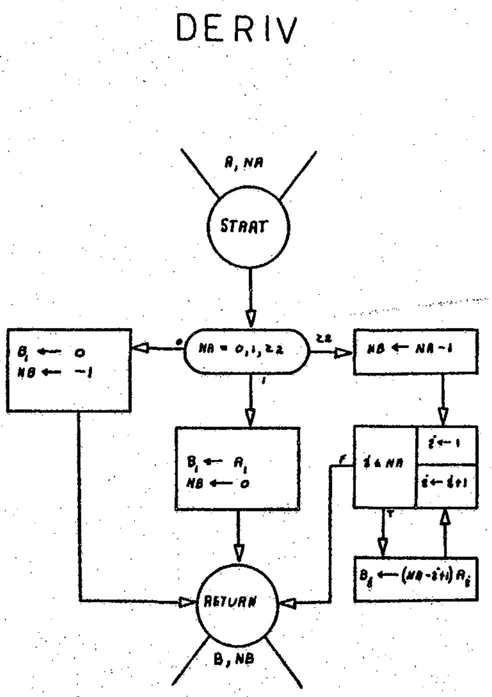


Figure 13.2. (Continued)



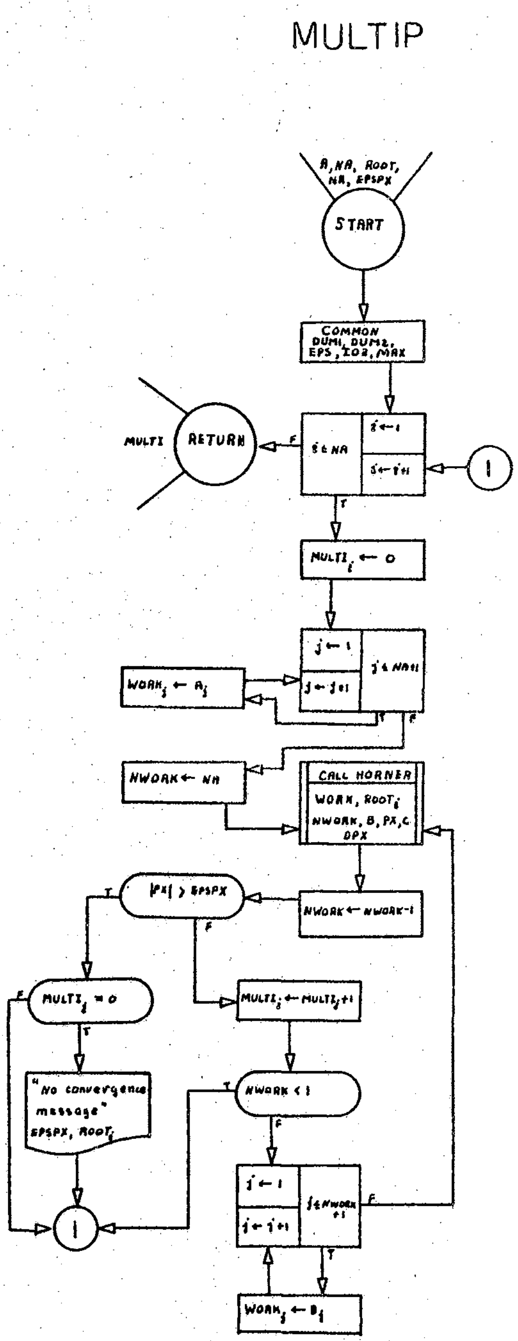
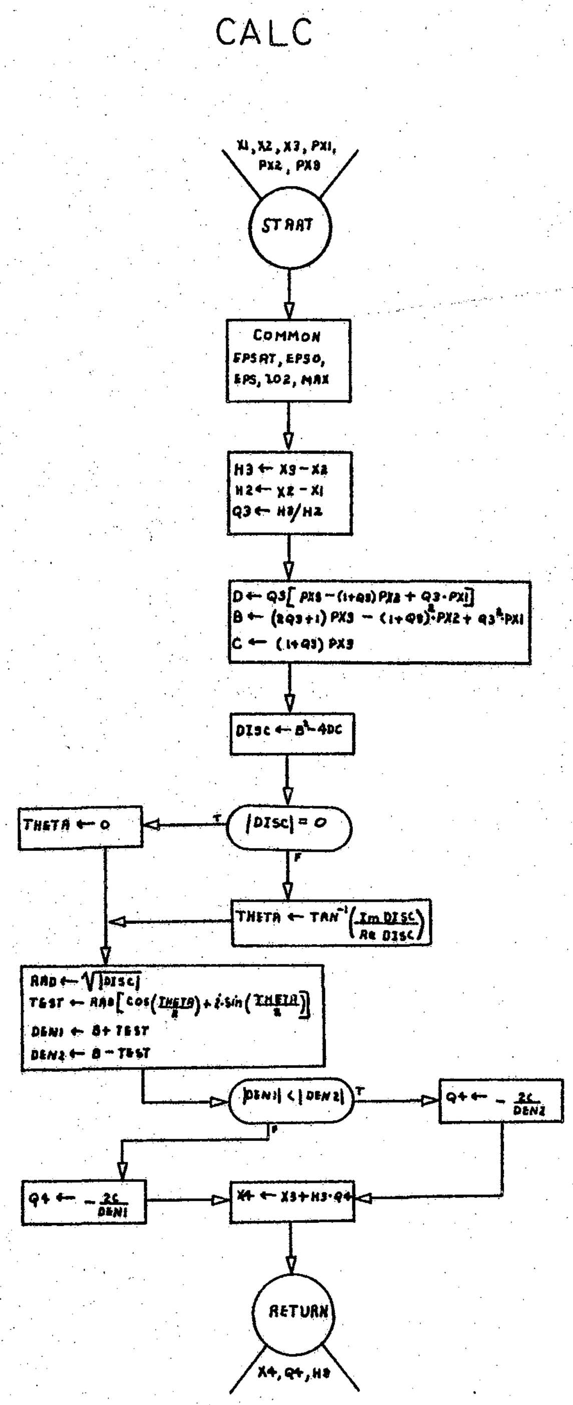


Figure 13.2. (Continued)



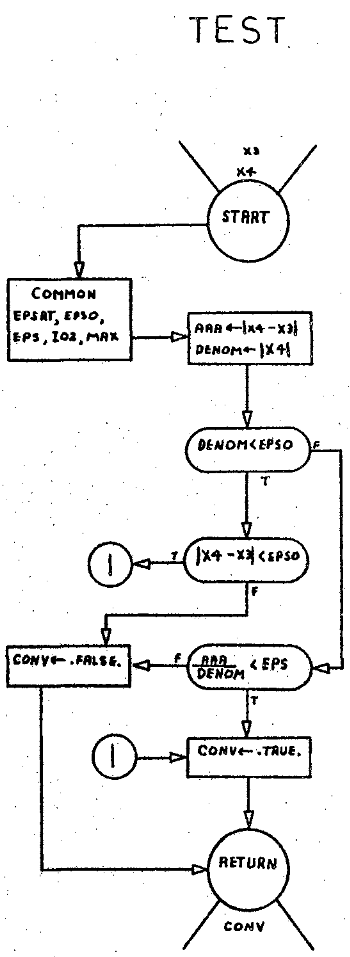
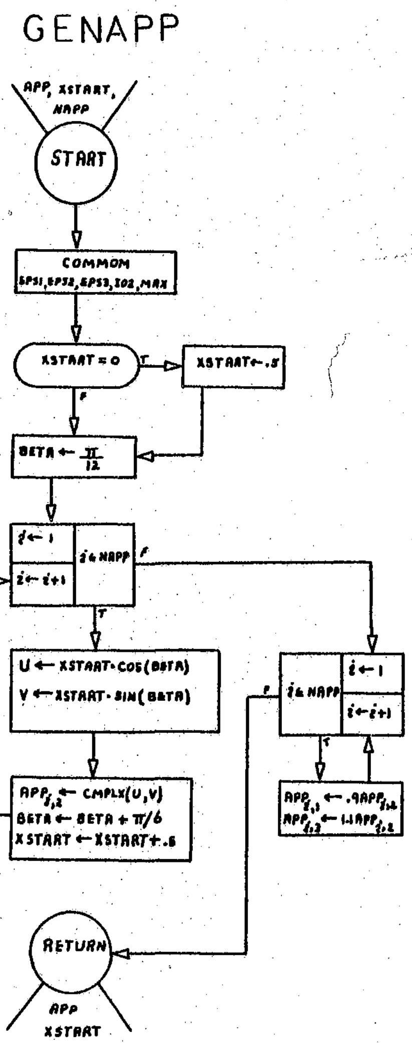


Figure 13.2. (Continued)



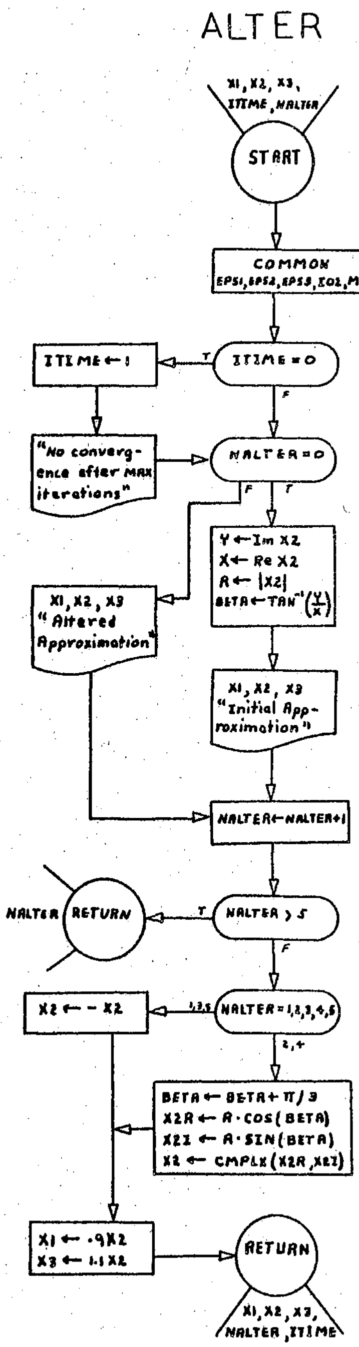


Figure 13.2. (Continued)

DIVIDE

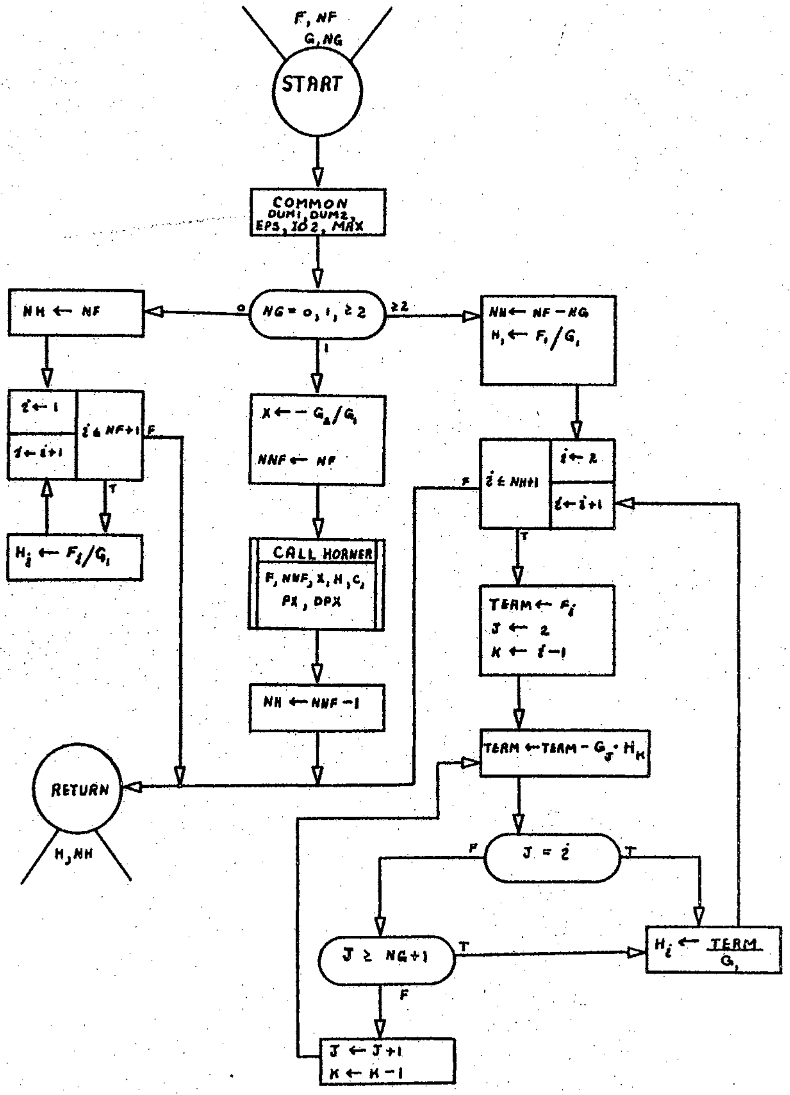
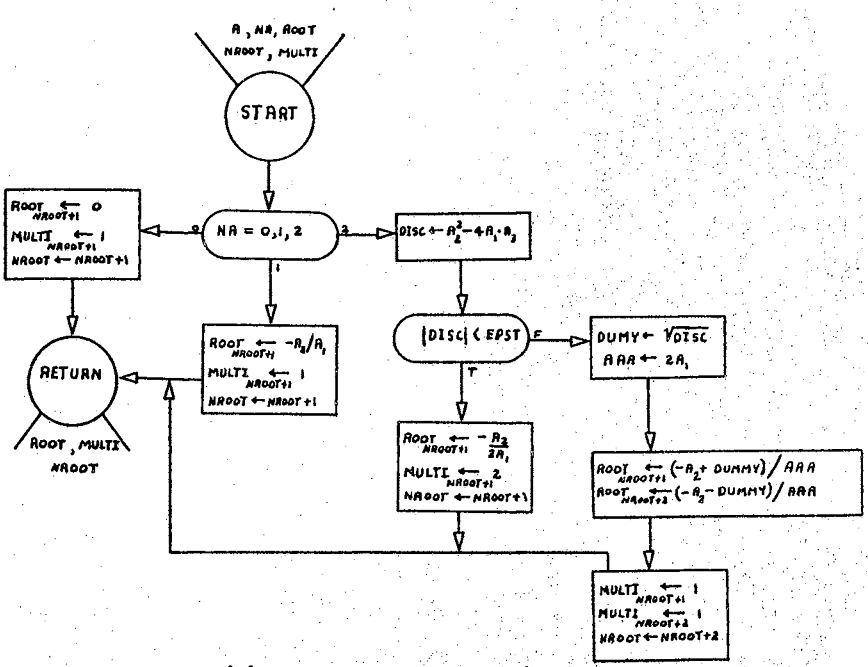


Figure 13.2. (Continued)

QUAD



COMSQT

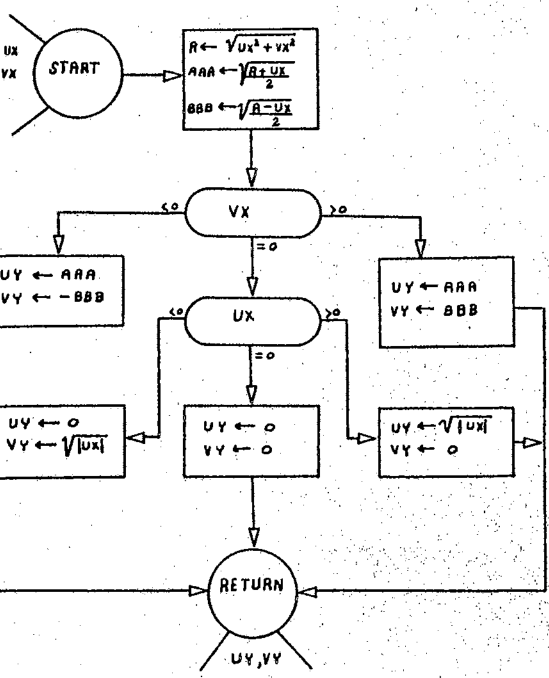


Figure 13.2. (Continued)

TABLE XXI-A

SINGLE PRECISION PROGRAM FOR G.C.D. - MULLER'S METHOD

```
$JOB 10414
C     ***********************************************************************
C     *                                                                     *
C     * SINGLE PRECISION PROGRAM FOR G.C.D. - MULLER'S METHOD               *
C     *                                                                     *
C     *                                                                     *
C     * THE G.C.D. METHOD EXTRACTS THE ZEROS AND THEIR MULTIPLICITIES OF A  *
C     * POLYNOMIAL OF MAXIMUM DEGREE 25.  ALL MULTIPLE ROOTS ARE REMOVED BY *
C     * DIVIDING THE POLYNOMIAL BY THE GREATEST COMMON DIVISOR OF THE POLYNOMIAL *
C     * AND ITS DERIVATIVE.  THE ZEROS OF THE RESULTING POLYNOMIAL ARE EXTRACTED *
C     * AND THEIR MULTIPLICITIES DETERMINED.                                *
C     *                                                                     *
C     ***********************************************************************
      COMPLEX P,OIAPP,ROOT,DP,RK,H
      DIMENSION P(26),OIAPP(25),ROOT(25),MULTI(25),DP(26),RK(26)
      DIMENSION H(26),ANAME(2)
      COMMON DUM1,EPSQ,EPSCNV,IO2,MAX
      DATA PNAME,HNAME,DNAME/2HP(,2HH(,2HD(/
      DATA ANAME(1),ANAME(2)/4HMULL,4HERS /
      IO1=5
      IO2=6
   10 NROOT=0
      READ(IO1,1000) NOPOLY,NP,NIAP,MAX,EPSG,EPSCNV,EPSQ,EPSMUL,XSTART,X
     1END,KCHECK
      IF(KCHECK.EQ.1) STOP
      KKK=NP+1
      READ(IO1,1010) (P(I),I=1,KKK)
      IF(NIAP.EQ.0) GO TO 20
      READ(IO1,1020) (OIAPP(I),I=1,NIAP)
   20 KKK=NP+1
      WRITE(IO2,1030) ANAME(1),ANAME(2),NOPOLY,NP
      WRITE(IO2,1035) (PNAME,I,P(I),I=1,KKK)
      WRITE(IO2,2060)
      WRITE(IO2,2000) NIAP
      WRITE(IO2,2010) MAX
      WRITE(IO2,2070) EPSG
      WRITE(IO2,2020) EPSCNV
      WRITE(IO2,2030) EPSMUL
      WRITE(IO2,2040) XSTART
      WRITE(IO2,2050) XEND
      IF(NP.GT.2) GO TO 90
      CALL QUAD(P,NP,ROOT,NROOT,MULTI,EPSQ)
   85 WRITE(IO2,1037)
      WRITE(IO2,1086) (I,ROOT(I),MULTI(I),I=1,NROOT)
      GO TO 10
   90 CALL DERIV(P,NP,DP,NDP)
      CALL GCD(P,NP,DP,NDP,RK,NRK,EPSG)
      CALL DIVIDE(P,NP,RK,NRK,H,NH)
      KKK=NH+1
      WRITE(IO2,1060)
      WRITE(IO2,1035) (HNAME,I,H(I),I=1,KKK)
      IF(NH.GT.2) GO TO 150
      CALL QUAD(H,NH,ROOT,NROOT,MULTI,EPSQ)
      CALL MULTIP(P,NP,ROOT,NROOT,MULTI,EPSMUL,NOPOLY)
      GO TO 85
  150 CALL MULLER(H,NH,OIAPP,NIAP,XSTART,XEND,EPSMUL,P,NP,NOPOLY)
      GO TO 10
 1086 FORMAT(2X,5HROOT(,I2,4H) = ,E14.7,3H + ,E14.7,2H I,10X,I2,11X,23HS
     1OLVED BY DIRECT METHOD)
```

TABLE XXI-A (Continued)

```
 1000 FORMAT(3(I2,1X),9X,I3,1X,4(E6.0,1X),13X,2(E7.0,1X),I1)
 1010 FORMAT(2E30.0)
 1020 FORMAT(2E30.0)
 1030 FORMAT(1H1,10X,41HGREATEST COMMON DIVISOR METHOD USED WITH ,2(A4),
     135HMETHOD TO FIND ZEROS OF POLYNOMIALS/11X,18HPOLYNOMIAL NUMBER ,I
     22,11H OF DEGREE ,I2////1X,28HTHE COEFFICIENTS OF P(X) ARE//)
 1035 FORMAT(3X,A2,I2,4H) = ,E14.7,3H + ,E14.7,2H I)
 2000 FORMAT(1X,41HNUMBER OF INITIAL APPROXIMATIONS GIVEN.   ,I2)
 2010 FORMAT(1X,29HMAXIMUM NUMBER OF ITERATIONS.,11X,I3)
 2020 FORMAT(1X,21HTEST FOR CONVERGENCE.,13X,E9.2)
 2030 FORMAT(1X,24HTEST FOR MULTIPLICITIES.,10X,E9.2)
 2040 FORMAT(1X,23HRADIUS TO START SEARCH.,11X,E9.2)
 2050 FORMAT(1X,21HRADIUS TO END SEARCH.,13X,E9.2)
 2060 FORMAT(//1X)
 2070 FORMAT(1X,31HTEST FOR ZERO IN SUBROUTINE GCD,3X,E9.2)
 1037 FORMAT(///,1X,13HZEROS OF P(X),37X,14HMULTIPLICITIES//)
 1060 FORMAT(///1X,42HTHE COEFFICIENTS OF H(X) = P(X)/G.C.D. ARE//)
      END
```

```
      SUBROUTINE DERIV(A,NA,B,NB)
C     **********************************************************************
C     *                                                                    *
C     * GIVEN A POLYNOMIAL P(X), SUBROUTINE DERIV COMPUTES THE COEFFICIENTS OF  *
C     * ITS DERIVATIVE P'(X).                                              *
C     *                                                                    *
C     **********************************************************************
      COMPLEX A,B
      DIMENSION A(26),B(26)
      IF(NA.GE.2) GO TO 30
      IF(NA.EQ.1) GO TO 20
      B(1)=0.0
      NB=-1
      GO TO 50
   20 B(1)=A(1)
      NB=0
      GO TO 50
   30 NB=NA-1
      DO 40 I=1,NA
      BBB=NA-I+1
   40 B(I)=BBB*A(I)
   50 RETURN
      END
```

TABLE XXI-A (Continued)

```
      SUBROUTINE DIVIDE(F,NF,G,NG,H,NH)
C   ****************************************************************************
C   *                                                                          *
C   * GIVEN TWO POLYNOMIALS F(X) AND G(X), SUBROUTINE DIVIDE COMPUTES THE      *
C   * QUOTIENT POLYNOMIAL  H(X) = F(X)/G(X).                                   *
C   *                                                                          *
C   ****************************************************************************
      COMPLEX F,G,H,X,PX,TERM
      DIMENSION F(26),G(26),H(26)
      COMMON DUM1,DUM2,EPS,IO2,MAX
      IF(NG.GE.2) GO TO 60
      IF(NG.EQ.1) GO TO 30
      NH=NF
      KKK=NF+1
      DO 20 I=1,KKK
   20 H(I)=F(I)/G(1)
      GO TO 100
   30 X=-G(2)/G(1)
      NNF=NF
      CALL HORNER(NNF,F,X,H,PX)
      NH=NNF-1
      GO TO 100
   60 NH=NF-NG
      H(1)=F(1)/G(1)
      KKK=NH+1
      DO 90 I=2,KKK
      TERM=F(I)
      J=2
      K=I-1
   70 TERM=TERM-G(J)*H(K)
      IF(J.EQ.I) GO TO 90
      IF(J.GE.NG+1) GO TO 90
      J=J+1
      K=K-1
      GO TO 70
   90 H(I)=TERM/G(1)
  100 RETURN
      END
```

TABLE XXI-A (Continued)

```
      SUBROUTINE GCD(P,NP,DP,NDP,RK,NK,EPSG)
C     ****************************************************************************
C     *                                                                          *
C     * GIVEN POLYNOMIALS P(X) AND DP(X) WHERE DEG. DP(X) IS LESS THAN DEG.      *
C     * P(X), SUBROUTINE GCD COMPUTES THE GREATEST COMMON DIVISOR OF P(X) AND    *
C     * DP(X).                                                                   *
C     *                                                                          *
C     ****************************************************************************
      COMPLEX R1,R2,RK,U,T,P,DP
      DIMENSION R1(26),R2(26),T(26),RK(26),P(26),DP(26)
      COMMON DUM1,DUM2,EPS,IO2,MAX
      N1=NP
      N2=NDP
      KKK=N1+1
      DO 10 I=1,KKK
   10 R1(I)=P(I)
      KKK=N2+1
      DO 20 I=1,KKK
   20 R2(I)=DP(I)
   50 U=R1(1)/R2(1)
      KKK=N2+1
      DO 70 I=1,KKK
   70 T(I)=R1(I)-U*R2(I)
      T(1)=0.0
      IF(N1.EQ.N2) GO TO 90
      KKK=N1+1
      NNN=N2+2
      DO 80 I=NNN,KKK
   80 T(I)=R1(I)
   90 KKK=N1+1
      DO 100 I=1,KKK
      BBB=CABS(T(I))
      IF(BBB.GT.EPSG) GO TO 120
  100 CONTINUE
      NK=N2
      KKK=NK+1
      DO 110 I=2,KKK
  110 RK(I)=R2(I)/R2(1)
      RK(1)=1.0
      GO TO 200
  120 M=N1+1-I
      IF(M.LT.N2) GO TO 160
      N1=M
      KKK=N1+1
      NNN=I-1
      DO 150 J=1,KKK
  150 R1(J)=T(NNN+J)
      GO TO 50
  160 N1=N2
      KKK=N1+1
      DO 170 J=1,KKK
  170 R1(J)=R2(J)
      N2=M
      KKK=N2+1
      NNN=I-1
      DO 180 J=1,KKK
  180 R2(J)=T(NNN+J)
      GO TO 50
  200 RETURN
      END
```

TABLE XXI-A (Continued)

```
      SUBROUTINE MULTIP(A,NA,ROOT,NR,MULTI,EPSPX)
C     ************************************************************************
C     *                                                                      *
C     * GIVEN NR ZEROS OF A POLYNOMIAL, SUBROUTINE MULTIP COMPUTES THEIR      *
C     * MULTIPLICITIES.                                                      *
C     *                                                                      *
C     ************************************************************************
      COMPLEX A,ROOT,WORK,PX,DPX,B
      DIMENSION A(26),ROOT(25),WORK(26),MULTI(25),B(26)
      COMMON DUM1,DUM2,EPS,IO2,MAX
      DO 50 I=1,NR
      MULTI(I)=0
      KKK=NA+1
      DO 20 J=1,KKK
   20 WORK(J)=A(J)
      NWORK=NA
   25 CALL HORNER(NWORK,WORK,ROOT(I),B,PX)
      NWORK=NWORK-1
      BBB=CABS(PX)
      IF(BBB.GT.EPSPX) GO TO 45
      MULTI(I)=MULTI(I)+1
      IF(NWORK.LT.1) GO TO 50
      KKK=NWORK+1
      DO 40 J=1,KKK
   40 WORK(J)=B(J)
      GO TO 25
   45 IF(MULTI(I).EQ.0) WRITE(IO2,1000) EPSPX,I,ROOT(I)
   50 CONTINUE
  100 RETURN
 1000 FORMAT(///14H THE EPSILON (,E14.7,49H) CHECK IN SUBROUTINE MULTIP
     1INDICATES THAT ROOT(,I2,4H) = ,E14.7,3H + ,E14.7,2H I/80H IS NOT C
     2LOSE ENOUGH TO BE A TRUE ROOT.  IT IS PRINTED BELOW WITH MULTIPLIC
     3ITY 0//)
      END
```

TABLE XXI-A (Continued)

```
      SUBROUTINE MULLER(A,NP,OIAPP,NAPP,XSTART,XEND,EPSM,P,NDEG,NOPOLY)
C     ***********************************************************************
C     *                                                                     *
C     * MULLER'S METHOD EXTRACTS THE ZEROS AND THEIR MULTIPLICITIES OF A    *
C     * POLYNOMIAL OF MAXIMUM DEGREE 25.  THROUGH THREE GIVEN POINTS THE    *
C     * POLYNOMIAL IS APPROXIMATED BY A QUADRATIC.  THE ZERO OF THE QUADRATIC *
C     * CLOSEST TO THE OLD APPROXIMATION IS TAKEN AS THE NEW APPROXIMATION.  *
C     * IN THIS MANNER A SEQUENCE IS OBTAINED CONVERGING TO A ZERO.         *
C     *                                                                     *
C     ***********************************************************************
      COMPLEX PX3,PX2,ROOT,X1,APP,X2,WORK,X3,B,X4,A,PX1,RAPP,PX4,H3,Q4
      COMPLEX OIAPP,P
      DIMENSION ROOT(25),MULT(25),APP(25,3),WORK(26),B(26),A(26),RAPP(25
     1,3)
      DIMENSION OIAPP(25),P(26)
      DATA PNAME,DNAME/2HP(,2HD(/
      LOGICAL CONV
      COMMON EPSRT,EPSO,EPS,IO2,MAX
      EPSRT=0.999
      NROOT=0
      IROOT=0
      IPATH=1
      NALTER=0
      ITIME=0
      NOMULT=0
      IAPP=1
      ITER=1
      IF(NAPP.NE.0) GO TO 18
      NAPP=NP
      CALL GENAPP(APP,NAPP,XSTART)
      GO TO 27
   18 DO 20 I=1,NAPP
   20 APP(I,2)=OIAPP(I)
      DO 25 I=1,NAPP
      APP(I,1)=0.9*APP(I,2)
   25 APP(I,3)=1.1*APP(I,2)
   27 KKK=NP+1
      DO 30 I=1,KKK
   30 WORK(I)=A(I)
      NWORK=NP
   40 X1=APP(IAPP,1)
      X2=APP(IAPP,2)
      X3=APP(IAPP,3)
      CALL HORNER(NWORK,WORK,X1,B,PX1)
      CALL HORNER(NWORK,WORK,X2,B,PX2)
      CALL HORNER(NWORK,WORK,X3,B,PX3)
   50 CALL CALC(X1,X2,X3,PX1,PX2,PX3,X4,Q4,H3)
   55 CALL HORNER(NWORK,WORK,X4,B,PX4)
      AB1=CABS(PX3)
      IF(AB1.EQ.0.0) GO TO 60
      QQQ=CABS(PX4)/CABS(PX3)
      IF(QQQ.LE.10.) GO TO 60
      Q4=0.5*Q4
      X4=X3+H3*Q4
      GO TO 55
   60 CALL TEST(X3,X4,CONV)
      IF(CONV) GO TO 120
      IF(ITER.LT.MAX) GO TO 110
      CALL ALTER(APP(IAPP,1),APP(IAPP,2),APP(IAPP,3),NALTER,ITIME)
      IF(NALTER.GT.5) GO TO 75
```

TABLE XXI-A (Continued)

```
      ITER=1
      GO TO 40
   75 IF(IAPP.LT.NAPP) GO TO 100
      IF(XEND.EQ.0.0) GO TO 70
      IF(XSTART.GT.XEND) GO TO 70
      NAPP=NP
      CALL GENAPP(APP,NAPP,XSTART)
      IAPP=0
      GO TO 100
   70 WRITE(IO2,1090)
      KKK=NWORK+1
      WRITE(IO2,1035) (DNAME,J,WORK(J),J=1,KKK)
   80 IF(NROOT.EQ.0) GO TO 90
      WRITE(IO2,1060)
      IF(IPATH.EQ.1) GO TO 82
   81 IPATH=2
      CALL BETTER(A,NP,ROOT,NROOT,RAPP,IROOT,MULT)
      WRITE(IO2,1200)
      CALL MULTIP(P,NDEG,ROOT,NROOT,MULT,EPSM)
   82 IF(NROOT.EQ.0)GO TO 90
      IF(IROOT.EQ.0) GO TO 85
      WRITE(IO2,1080)
      DO 83 I=1,IROOT
   83 WRITE(IO2,1085) I,ROOT(I),MULT(I),RAPP(I,2)
      IF(IROOT.LT.NROOT) GO TO 85
    ) GO TO 87
   85 KKK=IROOT+1
      WRITE(IO2,1086) (I,ROOT(I),MULT(I),I=KKK,NROOT)
   87 IF(IPATH.EQ.1) GO TO 81
      RETURN
   90 WRITE(IO2,1070) NOPOLY
      RETURN
  100 IAPP=IAPP+1
      ITER=1
      NALTER=0
      GO TO 40
  120 NROOT=NROOT+1
      IROOT=NROOT
      MULT(NROOT)=1
      NOMULT=NOMULT+1
      ROOT(NROOT)=X4
      RAPP(NROOT,1)=APP(IAPP,1)
      RAPP(NROOT,2)=APP(IAPP,2)
      RAPP(NROOT,3)=APP(IAPP,3)
  125 IF(NOMULT.LT.NP) GO TO 130
      GO TO 80
  130 CALL HORNER(NWORK,WORK,X4,B,PX4)
      NWORK=NWORK-1
      KKK=NWORK+1
      DO 140 I=1,KKK
  140 WORK(I)=B(I)
      CALL HORNER(NWORK,WORK,X4,B,PX4)
      CCC=CABS(PX4)
      IF(CCC.LT.EPSM) GO TO 150
      IF(NWORK.GT.2) GO TO 75
      IROOT=NROOT
      CALL QUAD(WORK,NWORK,ROOT,NROOT,MULT,EPSO)
      GO TO 80
  150 MULT(NROOT)=MULT(NROOT)+1
      NOMULT=NOMULT+1
```

TABLE XXI-A (Continued)

```
      GO TO 125
  110 X1=X2
      X2=X3
      X3=X4
      PX1=PX2
      PX2=PX3
      PX3=PX4
      ITER=ITER+1
      GO TO 50
 1090 FORMAT(///,1X,65HCOEFFICIENTS OF DEFLATED POLYNOMIAL FOR WHICH NO
     1ZEROS WERE FOUND//)
 1080 FORMAT(///1X,13HROOTS OF P(X),37X,14HMULTIPLICITIES,11X,21HINITIAL
     1 APPROXIMATION//)
 1070 FORMAT(///1X,42HNO ZEROS WERE FOUND FOR POLYNOMIAL NUMBER ,I2)
 1035 FORMAT(3X,A2,I2,4H) = ,E14.7,3H + ,E14.7,2H I)
 1085 FORMAT(2X,5HROOT(,I2,4H) = ,E14.7,3H + ,E14.7,2H I,10X,I2,10X,E14.
     17,3H + ,E14.7,2H I)
 1086 FORMAT(2X,5HROOT(,I2,4H) = ,E14.7,3H + ,E14.7,2H I,10X,I2,11X,23HS
     1OLVED BY DIRECT METHOD)
 1060 FORMAT(///35H BEFORE ATTEMPT TO IMPROVE ACCURACY)
 1200 FORMAT(///1X,37HAFTER THE ATTEMPT TO IMPROVE ACCURACY)
      END
```

TABLE XXI-A (Continued)

```
      SUBROUTINE BETTER(A,NP,ROOT,NROOT,RAPP,IROOT,MULT)
C   ***********************************************************************
C   *                                                                     *
C   * SUBROUTINE BETTER ATTEMPTS TO IMPROVE THE ACCURACY OF THE ZEROS FOUND *
C   * BY USING THEM AS INITIAL APPROXIMATIONS WITH MULLER'S METHOD APPLIED TO *
C   * THE FULL, UNDEFLATED POLYNOMIAL.                                    *
C   *                                                                     *
C   ***********************************************************************
      COMPLEX ROOT,A,BAPP,X1,X2,X3,PX1,PX2,PX3,B,ROOTS,X4,RAPP,Q4,H3
      LOGICAL CONV
      DIMENSION ROOT(25),A(26),BAPP(25,3),B(26),ROOTS(25),RAPP(25,3),MUL
     1T(25)
      COMMON EPSRT,EPSO,EPS,IO2,MAX
      IF(NROOT.LE.1) RETURN
      L=0
      DO 10 I=1,NROOT
      BAPP(I,1)=ROOT(I)*EPSRT
      BAPP(I,2)=ROOT(I)
   10 BAPP(I,3)=ROOT(I)*(2.0-EPSRT)
      DO 100 J=1,NROOT
      X1=BAPP(J,1)
      X2=BAPP(J,2)
      X3=BAPP(J,3)
      ITER=1
      CALL HORNER(NP,A,X1,B,PX1)
      CALL HORNER(NP,A,X2,B,PX2)
   20 CALL HORNER(NP,A,X3,B,PX3)
      CALL CALC(X1,X2,X3,PX1,PX2,PX3,X4,Q4,H3)
   30 CALL TEST(X3,X4,CONV)
      IF(CONV) GO TO 50
      IF(ITER.LT.MAX) GO TO 40
      WRITE(IO2,1000) J,ROOT(J),MAX
      WRITE(IO2,1010) X4
      IF(J.LT.IROOT) GO TO 33
      IF(J.EQ.IROOT) GO TO 35
      GO TO 100
   33 KKK=IROOT-1
      DO 34 K=J,KKK
      RAPP(K,1)=RAPP(K+1,1)
      RAPP(K,2)=RAPP(K+1,2)
   34 RAPP(K,3)=RAPP(K+1,3)
   35 IROOT=IROOT-1
      GO TO 100
   40 X1=X2
      X2=X3
      X3=X4
      PX1=PX2
      PX2=PX3
      ITER=ITER+1
      GO TO 20
   50 L=L+1
      ROOTS(L)=X4
      MULT(L)=MULT(J)
  100 CONTINUE
      IF(L.EQ.0) GO TO 120
      DO 110 I=1,L
  110 ROOT(I)=ROOTS(I)
      NROOT=L
      RETURN
  120 NROOT=0
```

TABLE XXI-A (Continued)

```
      RETURN
 1000 FORMAT(///42H IN THE ATTEMPT TO IMPROVE ACCURACY, ROOT(,I2,4H) = ,
     1E14.7,3H + ,E14.7,2H I,24H DID NOT CONVERGE AFTER ,I3,11H ITERATIO
     2NS)
 1010 FORMAT(30H THE PRESENT APPROXIMATION IS ,E14.7,3H + ,E14.7,2H I//)
      END
```

```
      SUBROUTINE CALC(X1,X2,X3,PX1,PX2,PX3,X4,Q4,H3)
C     ************************************************************************
C     *                                                                      *
C     * GIVEN THREE APPROXIMATIONS X(N-2), X(N-1), AND X(N), SUBROUTINE CALC *
C     * APPROXIMATES THE POLYNOMIAL BY A QUADRATIC AND SOLVES FOR THE ZERO OF*
C     * THE QUADRATIC CLOSEST TO X(N). THIS ZERO IS THE NEW APPROXIMATION    *
C     * X(N+1) TO THE ZERO OF THE POLYNOMIAL.                                *
C     *                                                                      *
C     ************************************************************************
      COMPLEX PX3,PX2,X1,X2,X3,PX1,H3,H2,Q3,D,B,C,DISC,DEN1,DEN2,Q4,X4
      COMPLEX TEST,CCC
      COMPLEX CMPLX
      COMMON EPSRT,EPSO,EPS,IO2,MAX
      H3=X3-X2
      H2=X2-X1
      Q3=H3/H2
      D=Q3*(PX3-((1.0+Q3)*PX2)+(Q3*PX1))
      B=(((2.0*Q3)+1.0)*PX3)-((1.0+Q3)*(1.0+Q3)*PX2)+(Q3*Q3*PX1)
      C=(1.0+Q3)*PX3
      DISC=(B*B)-(4.0*D*C)
      AAA=CABS(DISC)
      IF(AAA.EQ.0.0) GO TO 5
      GO TO 7
    5 THETA=0.0
      GO TO 9
    7 DISCI=AIMAG(DISC)
      DISCR=REAL(DISC)
      THETA=ATAN2(DISCI,DISCR)
    9 RAD=SQRT(AAA)
      ANGLE=THETA/2.0
      CCC=CMPLX(COS(ANGLE),SIN(ANGLE))
      TEST=RAD*CCC
      DEN1=B+TEST
      DEN2=B-TEST
      AAA=CABS(DEN1)
      BBB=CABS(DEN2)
      IF(AAA.LT.BBB) GO TO 10
      IF(AAA.EQ.0.0) GO TO 60
      Q4=(-2.0*C)/DEN1
      GO TO 50
   10 IF(BBB.EQ.0.0) GO TO 60
      Q4=(-2.0*C)/DEN2
      GO TO 50
   50 X4=X3+(H3*Q4)
      RETURN
   60 Q4=(1.0,0.0)
      GO TO 50
      END
```

TABLE XXI-A (Continued)

```
      SUBROUTINE GENAPP(APP,NAPP,XSTART)
C   ***************************************************************************
C   *                                                                         *
C   * SUBROUTINE GENAPP GENERATES N INITIAL APPROXIMATIONS, WHERE N IS THE    *
C   * DEGREE OF THE ORIGINAL POLYNOMIAL.                                      *
C   *                                                                         *
C   ***************************************************************************
      COMPLEX APP
      COMPLEX CMPLX
      DIMENSION APP(25,3)
      COMMON EPS1,EPS2,EPS3,IO2,MAX
      IF(XSTART.EQ.0.0) XSTART=0.5
      BETA=0.2617994
      DO 10 I=1,NAPP
      U=XSTART*COS(BETA)
      V=XSTART*SIN(BETA)
      APP(I,2)=CMPLX(U,V)
      BETA=BETA+0.5235988
   10 XSTART=XSTART+0.5
      DO 20 I=1,NAPP
      APP(I,1)=0.9*APP(I,2)
   20 APP(I,3)=1.1*APP(I,2)
      RETURN
      END
```

TABLE XXI-A (Continued)

```
      SUBROUTINE ALTER(X1,X2,X3,NALTER,ITIME)
C     ************************************************************************
C     *                                                                      *
C     * SUBROUTINE ALTER ALTERS THE INITIAL APPROXIMATIONS WHICH PRODUCE NO  *
C     * CONVERGENCE TO A ZERO. THIS IS DONE A MAXIMUM OF 5 TIMES FOR EACH ROOT. *
C     *                                                                      *
C     ************************************************************************
      COMPLEX X1,X2,X3
      COMPLEX CMPLX
      COMMON EPS1,EPS2,EPS3,IO2,MAX
      IF(ITIME.NE.0) GO TO 5
      ITIME=1
      WRITE(IO2,1010) MAX
    5 IF(NALTER.EQ.0) GO TO 10
      WRITE(IO2,1000) X1,X2,X3
      GO TO 20
   10 Y=AIMAG(X2)
      X=REAL(X2)
      R=CABS(X2)
      BETA=ATAN2(Y,X)
      WRITE(IO2,1020) X1,X2,X3
   20 NALTER=NALTER+1
      IF(NALTER.GT.5) RETURN
      GO TO (30,40,30,40,30),NALTER
   30 X2=-X2
      GO TO 50
   40 BETA=BETA+1.0471976
      X2R=R*COS(BETA)
      X2I=R*SIN(BETA)
      X2=CMPLX(X2R,X2I)
   50 X1=0.9*X2
      X3=1.1*X2
      RETURN
 1010 FORMAT(///1X,54HNO CONVERGENCE FOR THE FOLLOWING APPROXIMATIONS AF
     1TER ,I3,12H ITERATIONS.//)
 1000 FORMAT(1X,5HX1 = ,E14.7,3H + ,E14.7,2H I,10X,22HALTERED APPROXIMAT
     1IONS/1X,5HX2 = ,E14.7,3H + ,E14.7,2H I/1X,5HX3 = ,E14.7,3H + ,E14.
     27,2H I/)
 1020 FORMAT(1H0,5HX1 = ,E14.7,3H + ,E14.7,2H I,10X,22HINITIAL APPROXIMA
     1TIONS/1X,5HX2 = ,E14.7,3H + ,E14.7,2H I/1X,5HX3 = ,E14.7,3H + ,E14
     2.7,2H I/)
      END
```

TABLE XXI-A (Continued)

```
      SUBROUTINE TEST(X3,X4,CONV)
C   ****************************************************************************
C   *                                                                          *
C   * SUBROUTINE TEST CHECKS FOR CONVERGENCE OF THE SEQUENCE OF APPROX-      *
C   * IMATIONS BY TESTING THE EXPRESSION                                       *
C   * ABSOLUTE VALUE OF (X(N+1)-X(N))/ABSOLUTE VALUE OF X(N+1).                *
C   * WHEN IT IS AS SMALL AS DESIRED, CONVERGENCE IS OBTAINED.                 *
C   *                                                                          *
C   ****************************************************************************
      COMPLEX X3,X4
      LOGICAL CONV
      COMMON EPSRT,EPSO,EPS,IO2,MAX
      AAA=CABS(X4-X3)
      DENOM=CABS(X4)
      IF(DENOM.LT.EPSO) GO TO 20
      IF(AAA/DENOM.LT.EPS) GO TO 10
    5 CONV=.FALSE.
      GO TO 100
   10 CONV=.TRUE.
      GO TO 100
   20 IF(AAA.LT.EPSO) GO TO 10
      GO TO 5
  100 RETURN
      END
```

```
      SUBROUTINE HORNER(NA,A,X,B,PX)
C   ****************************************************************************
C   *                                                                          *
C   * HORNER'S METHOD COMPUTES THE VALUE OF THE POLYNOMIAL P(X) AT A POINT D.  *
C   * SYNTHETIC DIVISION IS USED TO DEFLATE THE POLYNOMIAL BY DIVIDING OUT THE *
C   * FACTOR (X-D).                                                            *
C   *                                                                          *
C   ****************************************************************************
      COMPLEX X,PX,B,A
      DIMENSION A(26),B(26)
      B(1)=A(1)
      NUM=NA+1
      DO 10 I=2,NUM
   10 B(I)=A(I)+B(I-1)*X
      PX=B(NUM)
      RETURN
      END
```

TABLE XXI-A (Continued)

```
      SUBROUTINE QUAD(A,NA,ROOT,NROOT,MULTI,EPST)
C     *****************************************************************************
C     *                                                                           *
C     * SUBROUTINE QUAD SOLVES DIRECTLY FOR THE ZEROS AND THEIR MULTIPLICITIES    *
C     * OF EITHER A QUADRATIC POLYNOMIAL OR A LINEAR FACTOR.  SOLUTION OF THE     *
C     * QUADRATIC IS DONE USING THE QUADRATIC FORMULA.                            *
C     *                                                                           *
C     *****************************************************************************
      COMPLEX A,DISC,ROOT,DUMMY,AAA
      COMPLEX CSQRT
      DIMENSION A(26),ROOT(25),MULTI(25)
      IF(NA.EQ.2) GO TO 7
      IF(NA.EQ.1) GO TO 5
      ROOT(NROOT+1)=0.0
      MULTI(NROOT+1)=1
      NROOT=NROOT+1
      GO TO 50
    5 ROOT(NROOT+1)=-A(2)/A(1)
      MULTI(NROOT+1)=1
      NROOT=NROOT+1
      GO TO 50
    7 DISC=A(2)*A(2)-(4.0*A(1)*A(3))
      BBB=CABS(DISC)
      IF(BBB.LT.EPST) GO TO 10
      DUMMY=CSQRT(DISC)
      AAA=2.0*A(1)
      ROOT(NROOT+1)=(-A(2)+DUMMY)/AAA
      ROOT(NROOT+2)=(-A(2)-DUMMY)/AAA
      MULTI(NROOT+1)=1
      MULTI(NROOT+2)=1
      NROOT=NROOT+2
      GO TO 50
   10 ROOT(NROOT+1)=(-A(2))/(2.0*A(1))
      MULTI(NROOT+1)=2
      NROOT=NROOT+1
   50 RETURN
      END
```

TABLE XXI-B

DOUBLE PRECISION PROGRAM FOR G.C.D. - MULLER'S METHOD

```
$JOB 10414
C     **************************************************************************
C     *                                                                        *
C     * DOUBLE PRECISION PROGRAM FOR G.C.D. - MULLER'S METHOD
C     *                                                                        *
C     *                                                                        *
C     * THE G.C.D. METHOD EXTRACTS THE ZEROS AND THEIR MULTIPLICITIES OF A     *
C     * POLYNOMIAL OF MAXIMUM DEGREE 25.  ALL MULTIPLE ROOTS ARE REMOVED BY    *
C     * DIVIDING THE POLYNOMIAL BY THE GREATEST COMMON DIVISOR OF THE POLYNOMIAL *
C     * AND ITS DERIVATIVE.  THE ZEROS OF THE RESULTING POLYNOMIAL ARE EXTRACTED *
C     * AND THEIR MULTIPLICITIES DETERMINED.                                    *
C     *                                                                        *
C     **************************************************************************
      DOUBLE PRECISION UP,VP,UOIAPP,VOIAPP,UROOT,VROOT,UDP,VDP,URK,VRK
      DOUBLE PRECISION UH,VH,EPSCHK,UDUMMY,VDUMMY,EPSQ,EPSMUL,EPSG,EPSCN
     1V,XSTART,XEND
      DOUBLE PRECISION DUM1
      DIMENSION ANAME(2),UP(26),VP(26),UOIAPP(25),VOIAPP(25),UROOT(25),V
     1ROOT(25),UDP(26),VDP(26),URK(26),VRK(26),UH(26),VH(26),MULTI(25)
      COMMON DUM1,EPSQ,EPSCNV,IO2,MAX
      DATA PNAME,HNAME,DNAME/2HP(,2HH(,2HD(/
      DATA ANAME(1),ANAME(2)/4HMULL,4HERS /
      IO1=5
      IO2=6
   10 NROOT=0
      READ(IO1,1000) NOPOLY,NP,NIAP,MAX,EPSG,EPSCNV,EPSQ,EPSMUL,XSTART,X
     1END,KCHECK
      IF(KCHECK.EQ.1) STOP
      KKK=NP+1
      READ(IO1,1010) (UP(I),VP(I),I=1,KKK)
      IF(NIAP.EQ.0) GO TO 20
      READ(IO1,1020) (UOIAPP(I),VOIAPP(I),I=1,NIAP)
   20 KKK=NP+1
      WRITE(IO2,1030) ANAME(1),ANAME(2),NOPOLY,NP
      WRITE(IO2,1035) (PNAME,I,UP(I),VP(I),I=1,KKK)
      WRITE(IO2,2060)
      WRITE(IO2,2000) NIAP
      WRITE(IO2,2010) MAX
      WRITE(IO2,2070) EPSG
      WRITE(IO2,2020) EPSCNV
      WRITE(IO2,2030) EPSMUL
      WRITE(IO2,2040) XSTART
      WRITE(IO2,2050) XEND
      IF(NP.GT.2) GO TO 90
      CALL QUAD(UP,VP,NP,UROOT,VROOT,MULTI,EPSQ)
   85 WRITE(IO2,1037)
      WRITE(IO2,1086) (I,UROOT(I),VROOT(I),MULTI(I),I=1,NROOT)
      GO TO 10
   90 CALL DERIV(UP,VP,NP,UDP,VDP,NDP)
      CALL GCD(UP,VP,NP,UDP,VDP,NDP,URK,VRK,NRK,EPSG)
      CALL DIVIDE(UP,VP,NP,URK,VRK,NRK,UH,VH,NH)
      KKK=NH+1
      WRITE(IO2,1060)
      WRITE(IO2,1035) (HNAME,I,UH(I),VH(I),I=1,KKK)
      IF(NH.GT.2) GO TO 150
      CALL QUAD(UH,VH,NH,UROOT,VROOT,NROOT,MULTI,EPSQ)
      CALL MULTIP(UP,VP,NP,UROOT,VROOT,NROOT,MULTI,EPSMUL)
      GO TO 85
  150 CALL MULLER(UH,VH,NH,UOIAPP,VOIAPP,NIAP,XSTART,XEND,EPSMUL,UP,VP,N
```

TABLE XXI-B (Continued)

```
     1P,NOPOLY)
      GO TO 10
 1000 FORMAT(3(I2,1X),9X,I3,1X,4(D6.0,1X),13X,2(D7.0,1X),I1)
 1010 FORMAT(2D30.0)
 1020 FORMAT(2D30.0)
 1030 FORMAT(1H1,10X,41HGREATEST COMMON DIVISOR METHOD USED WITH ,2(A4),
     135HMETHOD TO FIND ZEROS OF POLYNOMIALS/11X,18HPOLYNOMIAL NUMBER ,I
     22,11H OF DEGREE ,I2////1X,28HTHE COEFFICIENTS OF P(X) ARE//)
 1035 FORMAT(3X,A2,I2,4H) = ,D23.16,3H + ,D23.16,2H I)
 1037 FORMAT(///1X,13HZEROS OF P(X),55X,14HMULTIPLICITIES//)
 1086 FORMAT(3X,5HROOT(,I2,4H) = ,D23.16,3H + ,D23.16,2H I,7X,I2,8X,23HS
     1OLVED BY DIRECT METHOD)
 1060 FORMAT(///1X,42HTHE COEFFICIENTS OF H(X) = P(X)/G.C.D. ARE//)
 2000 FORMAT(1X,41HNUMBER OF INITIAL APPROXIMATIONS GIVEN.  ,I2)
 2010 FORMAT(1X,29HMAXIMUM NUMBER OF ITERATIONS.,11X,I3)
 2020 FORMAT(1X,21HTEST FOR CONVERGENCE.,13X,D9.2)
 2030 FORMAT(1X,24HTEST FOR MULTIPLICITIES.,10X,D9.2)
 2040 FORMAT(1X,23HRADIUS TO START SEARCH.,11X,D9.2)
 2050 FORMAT(1X,21HRADIUS TO END SEARCH.,13X,D9.2)
 2060 FORMAT(//1X)
 2070 FORMAT(1X,31HTEST FOR ZERO IN SUBROUTINE GCD,3X,D9.2)
      END
```

```
      SUBROUTINE DERIV(UA,VA,NA,UB,VB,NB)
C     ************************************************************************
C     *                                                                      *
C     * GIVEN A POLYNOMIAL P(X), SUBROUTINE DERIV COMPUTES THE COEFFICIENTS OF *
C     * ITS DERIVATIVE P'(X).                                                *
C     *                                                                      *
C     ************************************************************************
      DOUBLE PRECISION UA,VA,UB,VB
      DIMENSION UA(26),VA(26),UB(26),VB(26)
      IF(NA.GE.2) GO TO 30
      IF(NA.EQ.1) GO TO 20
      UB(1)=0.0
      VB(1)=0.0
      NB=-1
      GO TO 50
   20 UB(1)=UA(1)
      VB(1)=VA(1)
      NB=0
      GO TO 50
   30 NB=NA-1
      DO 40 I=1,NA
      BBB=NA-I+1
      UB(I)=BBB*UA(I)
   40 VB(I)=BBB*VA(I)
   50 RETURN
      END
```

TABLE XXI-B (Continued)

```
      SUBROUTINE DIVIDE(UF,VF,NF,UG,VG,NG,UH,VH,NH)
C   ************************************************************************
C   *                                                                      *
C   * GIVEN TWO POLYNOMIALS F(X) AND G(X), SUBROUTINE DIVIDE COMPUTES THE  *
C   * QUOTIENT POLYNOMIAL  H(X) = F(X)/G(X).                               *
C   *                                                                      *
C   ************************************************************************
      DOUBLE PRECISION UF,VF,UG,VG,UH,VH,UX,VX,UPX,VPX,UTERM,VTERM,UDUMM
     1Y,VDUMMY,EPS
      DOUBLE PRECISION DUM1,DUM2
      DOUBLE PRECISION DENOM
      DIMENSION UF(26),VF(26),UG(26),VG(26),UH(26),VH(26)
      COMMON DUM1,DUM2,EPS,IO2,MAX
      IF(NG.GE.2) GO TO 60
      IF(NG.EQ.1) GO TO 30
      NH=NF
      KKK=NF+1
      DO 20 I=1,KKK
      DENOM=UG(1)*UG(1)+VG(1)*VG(1)
      UH(I)=(UF(I)*UG(1)+VF(I)*VG(1))/DENOM
   20 VH(I)=(VF(I)*UG(1)-UF(I)*VG(1))/DENOM
      GO TO 100
   30 UDUMMY=-1.0*UG(2)
      VDUMMY=-1.0*VG(2)
      DENOM=UG(1)*UG(1)+VG(1)*VG(1)
      UX=(UDUMMY*UG(1)+VDUMMY*VG(1))/DENOM
      VX=(VDUMMY*UG(1)-UDUMMY*VG(1))/DENOM
      NNF=NF
      CALL HORNER(NNF,UF,VF,UX,VX,UH,VH,UPX,VPX)
      NH=NNF-1
      GO TO 100
   60 NH=NF-NG
      DENOM=UG(1)*UG(1)+VG(1)*VG(1)
      UH(1)=(UF(1)*UG(1)+VF(1)*VG(1))/DENOM
      VH(1)=(VF(1)*UG(1)-UF(1)*VG(1))/DENOM
      KKK=NH+1
      DO 95 I=2,KKK
      UTERM=UF(I)
      VTERM=VF(I)
      J=2
      K=I-1
   70 UTERM=UTERM-(UG(J)*UH(K)-VG(J)*VH(K))
      VTERM=VTERM-(VG(J)*UH(K)+UG(J)*VH(K))
      IF(J.EQ.I) GO TO 90
      IF(J.GE.NG+1) GO TO 90
      J=J+1
      K=K-1
      GO TO 70
   90 DENOM=UG(1)*UG(1)+VG(1)*VG(1)
      UH(I)=(UTERM*UG(1)+VTERM*VG(1))/DENOM
   95 VH(I)=(VTERM*UG(1)-UTERM*VG(1))/DENOM
  100 RETURN
      END
```

TABLE XXI-B (Continued)

```
      SUBROUTINE QUAD(UA,VA,NA,UROOT,VROOT,NROOT,MULTI,EPST)
C     ************************************************************************
C     *                                                                      *
C     * SUBROUTINE QUAD SOLVES DIRECTLY FOR THE ZEROS AND THEIR MULTIPLICITIES *
C     * OF EITHER A QUADRATIC POLYNOMIAL OR A LINEAR FACTOR.  SOLUTION OF THE  *
C     * QUADRATIC IS DONE USING THE QUADRATIC FORMULA.                       *
C     *                                                                      *
C     ************************************************************************
      DOUBLE PRECISION UA,VA,UROOT,VROOT,BBB,UAAA,VAAA,UDISC,VDISC,UDUMM
     1Y,VDUMMY,RDUMMY,SDUMMY,EPST,UBBB,VBBB
      DOUBLE PRECISION DSQRT
      DIMENSION UA(26),VA(26),UROOT(25),VROOT(25),MULTI(25)
      IF(NA.EQ.2) GO TO 7
      IF(NA.EQ.1) GO TO 5
      UROOT(NROOT+1)=0.0
      VROOT(NROOT+1)=0.0
      MULTI(NROOT+1)=1
      NROOT=NROOT+1
      GO TO 50
    5 BBB=UA(1)*UA(1)+VA(1)*VA(1)
      UROOT(NROOT+1)=(-UA(2)*UA(1)-VA(2)*VA(1))/BBB
      VROOT(NROOT+1)=(-VA(2)*UA(1)+UA(2)*VA(1))/BBB
      MULTI(NROOT+1)=1
      NROOT=NROOT+1
      GO TO 50
    7 UDISC=(UA(2)*UA(2)-VA(2)*VA(2))-(4.0*(UA(1)*UA(3)-VA(1)*VA(3)))
      VDISC=(VA(2)*UA(2)+UA(2)*VA(2))-(4.0*(VA(1)*UA(3)+UA(1)*VA(3)))
      BBB=DSQRT(UDISC*UDISC+VDISC*VDISC)
      IF(BBB.LT.EPST) GO TO 10
      CALL COMSQT(UDISC,VDISC,UDUMMY,VDUMMY)
      UBBB=-UA(2)+UDUMMY
      VBBB=-VA(2)+VDUMMY
      RDUMMY=-UA(2)-UDUMMY
      SDUMMY=-VA(2)-VDUMMY
      UAAA=2.0*UA(1)
      VAAA=2.0*VA(1)
      BBB=UAAA*UAAA+VAAA*VAAA
      UROOT(NROOT+1)=(UBBB*UAAA+VBBB*VAAA)/BBB
      VROOT(NROOT+1)=(VBBB*UAAA-UBBB*VAAA)/BBB
      UROOT(NROOT+2)=(RDUMMY*UAAA+SDUMMY*VAAA)/BBB
      VROOT(NROOT+2)=(SDUMMY*UAAA-RDUMMY*VAAA)/BBB
      MULTI(NROOT+1)=1
      MULTI(NROOT+2)=1
      NROOT=NROOT+2
      GO TO 50
   10 UAAA=2.0*UA(1)
      VAAA=2.0*VA(1)
      BBB=UAAA*UAAA+VAAA*VAAA
      UROOT(NROOT+1)=(-UA(2)*UAAA-VA(2)*VAAA)/BBB
      VROOT(NROOT+1)=(-VA(2)*UAAA+UA(2)*VAAA)/BBB
      MULTI(NROOT+1)=2
      NROOT=NROOT+1
   50 RETURN
      END
```

TABLE XXI-B (Continued)

```
      SUBROUTINE GCD(UP,VP,NP,UDP,VDP,NDP,URK,VRK,NK,EPSG)
C     *********************************************************************
C     *                                                                   *
C     * GIVEN POLYNOMIALS P(X) AND DP(X) WHERE DEG. DP(X) IS LESS THAN DEG.   *
C     * P(X), SUBROUTINE GCD COMPUTES THE GREATEST COMMON DIVISOR OF P(X) AND  *
C     * DP(X).                                                            *
C     *                                                                   *
C     *********************************************************************
      DOUBLE PRECISION EPSG,UP,VP,UDP,VDP,URK,VRK,UR1,VR1,UR2,VR2,UT,VT,
     1UU,VU,BBB,EPS
      DOUBLE PRECISION DUM1,DUM2
      DOUBLE PRECISION DENOM
      DOUBLE PRECISION DSQRT
      DIMENSION UR1(26),VR1(26),UR2(26),VR2(26),UT(26),VT(26),URK(26),VR
     1K(26),UP(26),VP(26),UDP(26),VDP(26)
      COMMON DUM1,DUM2,EPS,IO2,MAX
      N1=NP
      N2=NDP
      KKK=N1+1
      DO 10 I=1,KKK
      UR1(I)=UP(I)
   10 VR1(I)=VP(I)
      KKK=N2+1
      DO 20 I=1,KKK
      UR2(I)=UDP(I)
   20 VR2(I)=VDP(I)
   50 DENOM=UR2(1)*UR2(1)+VR2(1)*VR2(1)
      UU=(UR1(1)*UR2(1)+VR1(1)*VR2(1))/DENOM
      VU=(VR1(1)*UR2(1)-UR1(1)*VR2(1))/DENOM
      KKK=N2+1
      DO 70 I=1,KKK
      UT(I)=UR1(I)-(UU*UR2(I)-VU*VR2(I))
   70 VT(I)=VR1(I)-(VU*UR2(I)+UU*VR2(I))
      UT(1)=0.0
      VT(1)=0.0
      IF(N1.EQ.N2) GO TO 90
      KKK=N1+1
      NNN=N2+2
      DO 80 I=NNN,KKK
      UT(I)=UR1(I)
   80 VT(I)=VR1(I)
   90 KKK=N1+1
      DO 100 I=1,KKK
      BBB=DSQRT(UT(I)*UT(I)+VT(I)*VT(I))
      IF(BBB.GT.EPSG) GO TO 120
  100 CONTINUE
      NK=N2
      KKK=NK+1
      DO 110 I=2,KKK
      DENOM=UR2(1)*UR2(1)+VR2(1)*VR2(1)
      URK(I)=(UR2(I)*UR2(1)+VR2(I)*VR2(1))/DENOM
  110 VRK(I)=(VR2(I)*UR2(1)-UR2(I)*VR2(1))/DENOM
      URK(1)=1.0
      VRK(1)=0.0
      GO TO 200
  120 M=N1+1-I
      IF(M.LT.N2) GO TO 160
      N1=M
      KKK=N1+1
      NNN=I-1
```

TABLE XXI-B (Continued)

```
      DO 150 J=1,KKK
      UR1(J)=UT(NNN+J)
  150 VR1(J)=VT(NNN+J)
      GO TO 50
  160 N1=N2
      KKK=N1+1
      DO 170 J=1,KKK
      UR1(J)=UR2(J)
  170 VR1(J)=VR2(J)
      N2=M
      KKK=N2+1
      NNN=I-1
      DO 180 J=1,KKK
      UR2(J)=UT(NNN+J)
  180 VR2(J)=VT(NNN+J)
      GO TO 50
  200 RETURN
      END
```

```
      SUBROUTINE MULTIP(UA,VA,NA,UROOT,VROOT,NR,MULTI,EPSPX)
C    ****************************************************************************
C    *                                                                          *
C    * GIVEN NR ZEROS OF A POLYNOMIAL, SUBROUTINE MULTIP COMPUTES THEIR         *
C    * MULTIPLICITIES.                                                          *
C    *                                                                          *
C    ****************************************************************************
      DOUBLE PRECISION UA,VA,UROOT,VROOT,EPSPX,UWORK,VWORK,UB,VB
      DOUBLE PRECISION UPX,VPX,UDPX,VDPX,BBB,EPS
      DOUBLE PRECISION DUM1,DUM2
      DOUBLE PRECISION DSQRT
      DIMENSION UA(26),VA(26),UROOT(25),VROOT(25),UWORK(26),VWORK(26),MU
     1LTI(25),UB(26),VB(26)
      COMMON DUM1,DUM2,EPS,IO2,MAX
      DO 50 I=1,NR
      MULTI(I)=0
      KKK=NA+1
      DO 20 J=1,KKK
      UWORK(J)=UA(J)
   20 VWORK(J)=VA(J)
      NWORK=NA
   25 CALL HORNER(NWORK,UWORK,VWORK,UROOT(I),VROOT(I),UB,VB,UPX,VPX)
      NWORK=NWORK-1
      BBB=DSQRT(UPX*UPX+VPX*VPX)
      IF(BBB.GT.EPSPX) GO TO 45
      MULTI(I)=MULTI(I)+1
      IF(NWORK.LT.1) GO TO 50
      KKK=NWORK+1
      DO 40 J=1,KKK
      UWORK(J)=UB(J)
   40 VWORK(J)=VB(J)
      GO TO 25
   45 IF(MULTI(I).EQ.0) WRITE(IO2,1000) EPSPX,I,UROOT(I),VROOT(I)
   50 CONTINUE
  100 RETURN
 1000 FORMAT(///14H THE EPSILON (,D10.03,49H) CHECK IN SUBROUTINE MULTIP
     1 INDICATES THAT ROOT(,I2,4H) = ,D23.16,3H + ,D23.16,2H I/80H IS NO
     2T CLOSE ENOUGH TO BE A TRUE ROOT.  IT IS PRINTED BELOW WITH MULTIP
     3LICITY 0//)
      END
```

TABLE XXI-B (Continued)

```
      SUBROUTINE MULLER(UA,VA,NP,UOIAPP,VOIAPP,NAPP,XSTART,XEND,EPSM,UP,
     1VP,NDEG,NOPOLY)
C     ***********************************************************************
C     *                                                                     *
C     * MULLER'S METHOD EXTRACTS THE ZEROS AND THEIR MULTIPLICITIES OF A    *
C     * POLYNOMIAL OF MAXIMUM DEGREE 25.  THROUGH THREE GIVEN POINTS THE    *
C     * POLYNOMIAL IS APPROXIMATED BY A QUADRATIC.  THE ZERO OF THE QUADRATIC *
C     * CLOSEST TO THE OLD APPROXIMATION IS TAKEN AS THE NEW APPROXIMATION. *
C     * IN THIS MANNER A SEQUENCE IS OBTAINED CONVERGING TO A ZERO.         *
C     *                                                                     *
C     ***********************************************************************
      DOUBLE PRECISION UPX3,VPX3,UPX2,VPX2,UROOT,VROOT,UX1,VX1,UAPP,VAPP
     1,UX2,VX2,UWORK,VWORK,UX3,VX3,UB,VB,UX4,VX4,UA,VA,UPX1,VPX1,URAPP,V
     2RAPP,UPX4,VPX4,EPSRT,EPSO,EPS,CCC,EPSM,UH3,VH3,UQ4,VQ4,ABPX4,ABPX3
     3,QQQ,XSTART,XEND
      DOUBLE PRECISION DSQRT
      DOUBLE PRECISION UOIAPP,VOIAPP
      DOUBLE PRECISION UP,VP
      DIMENSION UROOT(25),VROOT(25),MULT(25),UAPP(25,3),VAPP(25,3),UWORK
     1(26),VWORK(26),UB(26),VB(26),UA(26),VA(26),URAPP(25,3),VRAPP(25,3)
      DIMENSION UOIAPP(25),VOIAPP(25)
      DIMENSION UP(26),VP(26)
      DATA PNAME,DNAME/2HP(,2HD(/
      LOGICAL CONV
      COMMON EPSRT,EPSO,EPS,IO2,MAX
      EPSRT=0.999
      NROOT=0
      IROOT=0
      IPATH=1
      NOMULT=0
      NALTER=0
      ITIME=0
      IAPP=1
      ITER=1
      IF(NAPP.NE.0) GO TO 18
      NAPP=NP
      CALL GENAPP(UAPP,VAPP,NAPP,XSTART)
      GO TO 27
   18 DO 20 I=1,NAPP
      UAPP(I,2)=UOIAPP(I)
   20 VAPP(I,2)=VOIAPP(I)
      DO 25 I=1,NAPP
      UAPP(I,1)=0.9*UAPP(I,2)
      VAPP(I,1)=0.9*VAPP(I,2)
      UAPP(I,3)=1.1*UAPP(I,2)
   25 VAPP(I,3)=1.1*VAPP(I,2)
   27 KKK=NP+1
      DO 30 I=1,KKK
      UWORK(I)=UA(I)
   30 VWORK(I)=VA(I)
      NWORK=NP
   40 UX1=UAPP(IAPP,1)
      VX1=VAPP(IAPP,1)
      UX2=UAPP(IAPP,2)
      VX2=VAPP(IAPP,2)
      UX3=UAPP(IAPP,3)
      VX3=VAPP(IAPP,3)
      CALL HORNER(NWORK,UWORK,VWORK,UX1,VX1,UB,VB,UPX1,VPX1)
      CALL HORNER(NWORK,UWORK,VWORK,UX2,VX2,UB,VB,UPX2,VPX2)
      CALL HORNER(NWORK,UWORK,VWORK,UX3,VX3,UB,VB,UPX3,VPX3)
```

TABLE XXI-B (Continued)

```
   50 CALL CALC(UX1,VX1,UX2,VX2,UX3,VX3,UPX1,VPX1,UPX2,VPX2,UPX3,VPX3,UX
     14,VX4,UQ4,VQ4,UH3,VH3)
   60 CALL HORNER(NWORK,UWORK,VWORK,UX4,VX4,UB,VB,UPX4,VPX4)
      ABPX4=DSQRT(UPX4*UPX4+VPX4*VPX4)
      ABPX3=DSQRT(UPX3*UPX3+VPX3*VPX3)
      IF(ABPX3.EQ.0.0) GO TO 70
      QQQ=ABPX4/ABPX3
      IF(QQQ.LE.10.) GO TO 70
      UQ4=0.5*UQ4
      VQ4=0.5*VQ4
      UX4=UX3+(UH3*UQ4-VH3*VQ4)
      VX4=VX3+(VH3*UQ4+UH3*VQ4)
      GO TO 60
   70 CALL TEST(UX3,VX3,UX4,VX4,CONV)
      IF(CONV) GO TO 120
      IF(ITER.LT.MAX) GO TO 110
      CALL ALTER(UAPP(IAPP,1),VAPP(IAPP,1),UAPP(IAPP,2),VAPP(IAPP,2),UAP
     1P(IAPP,3),VAPP(IAPP,3),NALTER,ITIME)
      IF(NALTER.GT.5) GO TO 75
      ITER=1
      GO TO 40
   75 IF(IAPP.LT.NAPP) GO TO 100
      IF(XEND.EQ.0.0) GO TO 77
      IF(XSTART.GT.XEND) GO TO 77
      NAPP=NP
      CALL GENAPP(UAPP,VAPP,NAPP,XSTART)
      IAPP=0
      GO TO 100
   77 WRITE(IO2,1090)
      KKK=NWORK+1
      WRITE(IO2,1035) (DNAME,J,UWORK(J),VWORK(J),J=1,KKK)
   80 IF(NROOT.EQ.0) GO TO 90
      WRITE(IO2,1060)
      IF(IPATH.EQ.1) GO TO 82
   81 IPATH=2
      CALL BETTER(UA,VA,NP,UROOT,VROOT,NROOT,URAPP,VRAPP,IROOT,MULT)
      WRITE(IO2,1200)
      CALL MULTIP(UP,VP,NDEG,UROOT,VROOT,NROOT,MULT,EPSM)
   82 IF(NROOT.EQ.0)GO TO 90
      IF(IROOT.EQ.0) GO TO 85
      WRITE(IO2,1080)
      DO 55 I=1,IROOT
   55 WRITE(IO2,1085) I,UROOT(I),VROOT(I),MULT(I),URAPP(I,2),VRAPP(I,2)
      IF(IROOT.LT.NROOT) GO TO 85
      GO TO 87
   85 KKK=IROOT+1
      WRITE(IO2,1086) (I,UROOT(I),VROOT(I),MULT(I),I=KKK,NROOT)
   87 IF(IPATH.EQ.1) GO TO 81
      RETURN
   90 WRITE(IO2,1070) NOPOLY
      RETURN
  100 IAPP=IAPP+1
      ITER=1
      NALTER=0
      GO TO 40
  120 NROOT=NROOT+1
      IROOT=NROOT
      MULT(NROOT)=1
      NOMULT=NOMULT+1
      UROOT(NROOT)=UX4
```

TABLE XXI-B (Continued)

```
      VROOT(NROOT)=VX4
      URAPP(NROOT,1)=UAPP(IAPP,1)
      VRAPP(NROOT,1)=VAPP(IAPP,1)
      URAPP(NROOT,2)=UAPP(IAPP,2)
      VRAPP(NROOT,2)=VAPP(IAPP,2)
      URAPP(NROOT,3)=UAPP(IAPP,3)
      VRAPP(NROOT,3)=VAPP(IAPP,3)
  125 IF(NOMULT.LT.NP) GO TO 130
      GO TO 80
  130 CALL HORNER(NWORK,UWORK,VWORK,UX4,VX4,UB,VB,UPX4,VPX4)
      NWORK=NWORK-1
      KKK=NWORK+1
      DO 140 I=1,KKK
      UWORK(I)=UB(I)
  140 VWORK(I)=VB(I)
      CALL HORNER(NWORK,UWORK,VWORK,UX4,VX4,UB,VB,UPX4,VPX4)
      CCC=DSQRT(UPX4*UPX4+VPX4*VPX4)
      IF(CCC.LT.EPSM) GO TO 150
      IF(NWORK.GT.2) GO TO 75
      IROOT=NROOT
      CALL QUAD(UWORK,VWORK,NWORK,UROOT,VROOT,NROOT,MULT,EPSO)
      GO TO 80
  150 MULT(NROOT)=MULT(NROOT)+1
      NOMULT=NOMULT+1
      GO TO 125
  110 UX1=UX2
      VX1=VX2
      UX2=UX3
      VX2=VX3
      UX3=UX4
      VX3=VX4
      UPX1=UPX2
      VPX1=VPX2
      UPX2=UPX3
      VPX2=VPX3
      UPX3=UPX4
      VPX3=VPX4
      ITER=ITER+1
      GO TO 50
 1090 FORMAT(///,1X,65HCOEFFICIENTS OF DEFLATED POLYNOMIAL FOR WHICH NO
     1ZEROS WERE FOUND//)
 1080 FORMAT(///1X,13HROOTS OF P(X),52X,14HMULTIPLICITIES,17X,21HINITIAL
     1 APPROXIMATION//)
 1070 FORMAT(//,43H NO ZEROS WERE FOUND FOR POLYNOMIAL NUMBER ,I2)
 1086 FORMAT(2X,5HROOT(,I2,4H) = ,D23.16,3H + ,D23.16,2H I,8X,I2,9X,23HS
     1OLVED BY DIRECT METHOD)
 1035 FORMAT(3X,A2,I2,4H) = ,D23.16,3H + ,D23.16,2H I)
 1085 FORMAT(2X,5HROOT(,I2,4H) = ,D23.16,3H + ,D23.16,2H I,8X,I2,8X,D23.
     116,3H + ,D23.16,2H I)
 1000 FORMAT(3(I2,1X),9X,I3,8X,3(D6.0,1X),13X,2(D7.0,1X),I1)
 1060 FORMAT(///35H BEFORE ATTEMPT TO IMPROVE ACCURACY)
 1200 FORMAT(///1X,37HAFTER THE ATTEMPT TO IMPROVE ACCURACY)
      END
```

TABLE XXI-B (Continued)

```
      SUBROUTINE ALTER(X1R,X1I,X2R,X2I,X3R,X3I,NALTER,ITIME)
C     ************************************************************************
C     *                                                                      *
C     * SUBROUTINE ALTER ALTERS THE INITIAL APPROXIMATIONS WHICH PRODUCE NO  *
C     * CONVERGENCE TO A ZERO. THIS IS DONE A MAXIMUM OF 5 TIMES FOR EACH ROOT. *
C     *                                                                      *
C     ************************************************************************
      DOUBLE PRECISION X1R,X1I,X2R,X2I,X3R,X3I,EPS1,EPS2,EPS3,R,BETA
      DOUBLE PRECISION DCOS,DSIN
      DOUBLE PRECISION DATAN2
      DOUBLE PRECISION DSQRT
      COMMON EPS1,EPS2,EPS3,IO2,MAX
      IF(ITIME.NE.0) GO TO 5
      ITIME=1
      WRITE(IO2,1010) MAX
    5 IF(NALTER.EQ.0) GO TO 10
      WRITE(IO2,1000) X1R,X1I,X2R,X2I,X3R,X3I
      GO TO 20
   10 R=DSQRT(X2R*X2R+X2I*X2I)
      BETA=DATAN2(X2I,X2R)
      WRITE(IO2,1020) X1R,X1I,X2R,X2I,X3R,X3I
   20 NALTER=NALTER+1
      IF(NALTER.GT.5) RETURN
      GO TO (30,40,30,40,30),NALTER
   30 X2R=-X2R
      X2I=-X2I
      GO TO 50
   40 BETA=BETA+1.0471976
      X2R=R*DCOS(BETA)
      X2I=R*DSIN(BETA)
   50 X1R=0.9*X2R
      X1I=0.9*X2I
      X3R=1.1*X2R
      X3I=1.1*X2I
      RETURN
 1000 FORMAT(1X,5HX1 = ,D23.16,3H + ,D23.16,2H I,10X,22HALTERED APPROXIM
     1ATIONS/1X,5HX2 = ,D23.16,3H + ,D23.16,2H I/1X,5HX3 = ,D23.16,3H +
     2,D23.16,2H I/)
 1020 FORMAT(1H0,5HX1 = ,D23.16,3H + ,D23.16,2H I,10X,22HINITIAL APPROXI
     1MATIONS/1X,5HX2 = ,D23.16,3H + ,D23.16,2H I/1X,5HX3 = ,D23.16,3H +
     2 ,D23.16,2H I/)
 1010 FORMAT(///1X,54HNO CONVERGENCE FOR THE FOLLOWING APPROXIMATIONS AF
     1TER ,I3,12H ITERATIONS.//)
      END
```

TABLE XXI-B (Continued)

```
      SUBROUTINE GENAPP(APPR,APPI,NAPP,XSTART)
C     ************************************************************************
C     *                                                                      *
C     * SUBROUTINE GENAPP GENERATES N INITIAL APPROXIMATIONS, WHERE N IS THE. *
C     * DEGREE OF THE ORIGINAL POLYNOMIAL.                                   *
C     *                                                                      *
C     ************************************************************************
      DOUBLE PRECISION APPR,APPI,XSTART,EPS1,EPS2,EPS3,BETA
      DOUBLE PRECISION DCOS,DSIN
      DIMENSION APPR(25,3),APPI(25,3)
      COMMON EPS1,EPS2,EPS3,IO2,MAX
      IF(XSTART.EQ.0.0) XSTART=0.5
      BETA=0.2617994
      DO 10 I=1,NAPP
      APPR(I,2)=XSTART*DCOS(BETA)
      APPI(I,2)=XSTART*DSIN(BETA)
      BETA=BETA+0.5235988
   10 XSTART=XSTART+0.5
      DO 20 I=1,NAPP
      APPR(I,1)=0.9*APPR(I,2)
      APPI(I,1)=0.9*APPI(I,2)
      APPR(I,3)=1.1*APPR(I,2)
   20 APPI(I,3)=1.1*APPI(I,2)
      RETURN
      END
```

TABLE XXI-B (Continued)

```
      SUBROUTINE BETTER(UA,VA,NP,UROOT,VROOT,NROOT,URAPP,VRAPP,IROOT,MUL
     1T)
C     ***********************************************************************
C     *                                                                     *
C     * SUBROUTINE BETTER ATTEMPTS TO IMPROVE THE ACCURACY OF THE ZEROS FOUND *
C     * BY USING THEM AS INITIAL APPROXIMATIONS WITH MULLER'S METHOD APPLIED TO *
C     * THE FULL, UNDEFLATED POLYNOMIAL.                                    *
C     *                                                                     *
C     ***********************************************************************
      DOUBLE PRECISION UROOT,VROOT,UA,VA,UBAPP,VBAPP,UX1,VX1,UX2,VX2,UX3
     1,VX3,UPX1,VPX1,UPX2,VPX2,UPX3,VPX3,UB,VB,UROOTS,VROOTS,EPSRT,UX4,V
     2X4,URAPP,VRAPP,EPSO,EPS,UQ4,VQ4,UH3,VH3
      LOGICAL CONV
      DIMENSION UROOT(25),VROOT(25),UA(26),VA(26),UBAPP(25,3),VBAPP(25,3
     1),UB(26),VB(26),UROOTS(25),VROOTS(25),URAPP(25,3),VRAPP(25,3),MULT
     3(25)
      COMMON EPSRT,EPSO,EPS,IO2,MAX
      IF(NROOT.LE.1) RETURN
      L=0
      DO 10 I=1,NROOT
      UBAPP(I,1)=UROOT(I)*EPSRT
      VBAPP(I,1)=VROOT(I)*EPSRT
      UBAPP(I,2)=UROOT(I)
      VBAPP(I,2)=VROOT(I)
      UBAPP(I,3)=UROOT(I)*(2.0-EPSRT)
   10 VBAPP(I,3)=VROOT(I)*(2.0-EPSRT)
      DO 100 J=1,NROOT
      UX1=UBAPP(J,1)
      VX1=VBAPP(J,1)
      UX2=UBAPP(J,2)
      VX2=VBAPP(J,2)
      UX3=UBAPP(J,3)
      VX3=VBAPP(J,3)
      ITER=1
      CALL HORNER(NP,UA,VA,UX1,VX1,UB,VB,UPX1,VPX1)
      CALL HORNER(NP,UA,VA,UX2,VX2,UB,VB,UPX2,VPX2)
   20 CALL HORNER(NP,UA,VA,UX3,VX3,UB,VB,UPX3,VPX3)
      CALL CALC(UX1,VX1,UX2,VX2,UX3,VX3,UPX1,VPX1,UPX2,VPX2,UPX3,VPX3,UX
     14,VX4,UQ4,VQ4,UH3,VH3)
   30 CALL TEST(UX3,VX3,UX4,VX4,CONV)
      IF(CONV) GO TO 50
      IF(ITER.LT.MAX) GO TO 40
      WRITE(IO2,1000) J,UROOT(J),VROOT(J),MAX
      WRITE(IO2,1010) UX4,VX4
      IF(J.LT.IROOT) GO TO 33
      IF(J.EQ.IROOT) GO TO 35
      GO TO 100
   33 KKK=IROOT-1
      DO 34 K=J,KKK
      URAPP(K,1)=URAPP(K+1,1)
      VRAPP(K,1)=VRAPP(K+1,1)
      URAPP(K,2)=URAPP(K+1,2)
      VRAPP(K,2)=VRAPP(K+1,2)
      URAPP(K,3)=URAPP(K+1,3)
   34 VRAPP(K,3)=VRAPP(K+1,3)
   35 IROOT=IROOT-1
      GO TO 100
   40 UX1=UX2
      VX1=VX2
      UX2=UX3
```

TABLE XXI-B (Continued)

```
      VX2=VX3
      UX3=UX4
      VX3=VX4
      UPX1=UPX2
      VPX1=VPX2
      UPX2=UPX3
      VPX2=VPX3
      ITER=ITER+1
      GO TO 20
   50 L=L+1
      UROOTS(L)=UX4
      VROOTS(L)=VX4
      MULT(L)=MULT(J)
  100 CONTINUE
      IF(L.EQ.0) GO TO 120
      DO 110 I=1,L
      UROOT(I)=UROOTS(I)
  110 VROOT(I)=VROOTS(I)
      NROOT=L
      RETURN
  120 NROOT=0
      RETURN
 1000 FORMAT(///42H IN THE ATTEMPT TO IMPROVE ACCURACY, ROOT(,I2,4H) = ,
     1D23.16,3H + ,D23.16,2H I/24H DID NOT CONVERGE AFTER ,I3,11H ITERAT
     2IONS)
 1010 FORMAT(30H THE PRESENT APPROXIMATION IS ,D23.16,3H + ,D23.16,2H I/
     1/)
      END
```

TABLE XXI-B (Continued)

```
      SUBROUTINE CALC(UX1,VX1,UX2,VX2,UX3,VX3,UPX1,VPX1,UPX2,VPX2,UPX3,V
     1PX3,UX4,VX4,UQ4,VQ4,UH3,VH3)
C    ***********************************************************************
C    *                                                                     *
C    * GIVEN THREE APPROXIMATIONS X(N-2), X(N-1), AND X(N), SUBROUTINE CALC *
C    * APPROXIMATES THE POLYNOMIAL BY A QUADRATIC AND SOLVES FOR THE ZERO OF *
C    * THE QUADRATIC CLOSEST TO X(N). THIS ZERO IS THE NEW APPROXIMATION   *
C    * X(N+1) TO THE ZERO OF THE POLYNOMIAL.                               *
C    *                                                                     *
C    ***********************************************************************
      DOUBLE PRECISION ARG1,ARG2
      DOUBLE PRECISION UPX3,VPX3,UPX2,VPX2,UX1,VX1,UX2,VX2,UX3,VX3,UPX1,
     1VPX1,UH3,VH3,UH2,VH2,UQ3,VQ3,UD,VD,UB,VB,UC,VC,UDISC,VDISC,UCCC,VC
     2CC,UDEN1,VDEN1,UDEN2,VDEN2,UQ4,VQ4,UX4,VX4,EPSRT,EPSO,EPS,UDDD,VDD
     3D,AAA,BBB,RAD,UAAA,VAAA,UBBB,VBBB
      DOUBLE PRECISION THETA,ANGLE,UTEST,VTEST
      DOUBLE PRECISION DATAN2,DCOS,DSIN,DSQRT
      COMMON EPSRT,EPSO,EPS,IO2,MAX
      UH3=UX3-UX2
      VH3=VX3-VX2
      UH2=UX2-UX1
      VH2=VX2-VX1
      BBB=UH2*UH2+VH2*VH2
      UQ3=(UH3*UH2+VH3*VH2)/BBB
      VQ3=(VH3*UH2-UH3*VH2)/BBB
      UDDD=1.0+UQ3
      VDDD=VQ3
      UD=(UPX3-(UDDD*UPX2-VDDD*VPX2))+(UQ3*UPX1-VQ3*VPX1)
      VD=(VPX3-(VDDD*UPX2+UDDD*VPX2))+(VQ3*UPX1+UQ3*VPX1)
      UAAA=2.0*UQ3
      VAAA=2.0*VQ3
      UAAA=UAAA+1.0
      UBBB=UDDD*UDDD-VDDD*VDDD
      VBBB=VDDD*UDDD+UDDD*VDDD
      UCCC=UQ3*UQ3-VQ3*VQ3
      VCCC=VQ3*UQ3+UQ3*VQ3
      UB=((UAAA*UPX3-VAAA*VPX3)-(UBBB*UPX2-VBBB*VPX2))+(UCCC*UPX1-VCCC*V
     1PX1)
      VB=((VAAA*UPX3+UAAA*VPX3)-(VBBB*UPX2+UBBB*VPX2))+(VCCC*UPX1+UCCC*V
     1PX1)
      UC=UDDD*UPX3-VDDD*VPX3
      VC=VDDD*UPX3+UDDD*VPX3
      UDISC=(UB*UB-VB*VB)-(4.0*(UD*UC-VD*VC))
      VDISC=(2.0*(VB*UB))-(4.0*(VD*UC+UD*VC))
      AAA=DSQRT(UDISC*UDISC+VDISC*VDISC)
      IF(AAA.EQ.0.0) GO TO 5
      GO TO 7
    5 THETA=0.0
      GO TO 9
    7 THETA=DATAN2(VDISC,UDISC)
    9 RAD=DSQRT(AAA)
      ANGLE=THETA/2.0
      UTEST=RAD*DCOS(ANGLE)
      VTEST=RAD*DSIN(ANGLE)
      UDEN1=UB+UTEST
      VDEN1=VB+VTEST
      UDEN2=UB-UTEST
      VDEN2=VB-VTEST
      ARG1=UDEN1*UDEN1+VDEN1*VDEN1
      ARG2=UDEN2*UDEN2+VDEN2*VDEN2
```

TABLE XXI-B (Continued)

```
      AAA=DSQRT(ARG1)
      BBB=DSQRT(ARG2)
      IF(AAA.LT.BBB) GO TO 10
      IF(AAA.EQ.0.0) GO TO 60
      UAAA=-2.0*UC
      VAAA=-2.0*VC
      UQ4=(UAAA*UDEN1+VAAA*VDEN1)/ARG1
      VQ4=(VAAA*UDEN1-UAAA*VDEN1)/ARG1
      GO TO 50
   10 IF(BBB.EQ.0.0) GO TO 60
      UAAA=-2.0*UC
      VAAA=-2.0*VC
      UQ4=(UAAA*UDEN2+VAAA*VDEN2)/ARG2
      VQ4=(VAAA*UDEN2-UAAA*VDEN2)/ARG2
      GO TO 50
   50 UX4=UX3+(UH3*UQ4-VH3*VQ4)
      VX4=VX3+(VH3*UQ4+UH3*VQ4)
      RETURN
   60 UQ4=1.0
      VQ4=0.0
      GO TO 50
      END
```

```
      SUBROUTINE TEST(UX3,VX3,UX4,VX4,CONV)
C    ***********************************************************************
C    *                                                                     *
C    * SUBROUTINE TEST CHECKS FOR CONVERGENCE OF THE SEQUENCE OF APPROX-   *
C    * IMATIONS BY TESTING THE EXPRESSION                                  *
C    * ABSOLUTE VALUE OF (X(N+1)-X(N))/ABSOLUTE VALUE OF X(N+1).          *
C    * WHEN IT IS AS SMALL AS DESIRED, CONVERGENCE IS OBTAINED.            *
C    *                                                                     *
C    ***********************************************************************
      DOUBLE PRECISION UX3,VX3,UX4,VX4,EPSRT,EPSO,EPS,AAA,UDUMMY,VDUMMY,
     1DENOM
      DOUBLE PRECISION DSQRT
      LOGICAL CONV
      COMMON EPSRT,EPSO,EPS,IO2,MAX
      UDUMMY=UX4-UX3
      VDUMMY=VX4-VX3
      AAA=DSQRT(UDUMMY*UDUMMY+VDUMMY*VDUMMY)
      DENOM=DSQRT(UX4*UX4+VX4*VX4)
      IF(DENOM.LT.EPSO) GO TO 20
      IF(AAA/DENOM.LT.EPS) GO TO 10
    5 CONV=.FALSE.
      GO TO 100
   10 CONV=.TRUE.
      GO TO 100
   20 IF(AAA.LT.EPSO) GO TO 10
      GO TO 5
  100 RETURN
      END
```

TABLE XXI-B (Continued)

```
      SUBROUTINE HORNER(NA,UA,VA,UX,VX,UB,VB,UPX,VPX)
C    **********************************************************************
C    *                                                                    *
C    * HORNER'S METHOD COMPUTES THE VALUE OF THE POLYNOMIAL P(X) AT A POINT D.  *
C    * SYNTHETIC DIVISION IS USED TO DEFLATE THE POLYNOMIAL BY DIVIDING OUT THE *
C    * FACTOR (X-D).                                                      *
C    *                                                                    *
C    **********************************************************************
      DOUBLE PRECISION UX,VX,UPX,VPX,UB,VB,UA,VA
      DIMENSION UA(26),VA(26),UB(26),VB(26)
      UB(1)=UA(1)
      VB(1)=VA(1)
      NUM=NA+1
      DO 10 I=2,NUM
      UB(I)=UA(I)+(UB(I-1)*UX-VB(I-1)*VX)
   10 VB(I)=VA(I)+(VB(I-1)*UX+UB(I-1)*VX)
      UPX=UB(NUM)
      VPX=VB(NUM)
      RETURN
      END
```

```
      SUBROUTINE COMSQT(UX,VX,UY,VY)
C    **********************************************************************
C    *                                                                    *
C    * THIS SUBROUTINE COMPUTES THE SQUARE ROOT OF A COMPLEX NUMBER.      *
C    *                                                                    *
C    **********************************************************************
      DOUBLE PRECISION UX,VX,UY,VY,DUMMY,R,AAA,BBB
      DOUBLE PRECISION DSQRT,DABS
      R=DSQRT(UX*UX+VX*VX)
      AAA=DSQRT(DABS((R+UX)/2.0))
      BBB=DSQRT(DABS((R-UX)/2.0))
      IF(VX) 10,20,30
   10 UY=AAA
      VY=-1.0*BBB
      GO TO 100
   20 IF(UX) 40,50,60
   30 UY=AAA
      VY=BBB
      GO TO 100
   40 DUMMY=DABS(UX)
      UY=0.0
      VY=DSQRT(DUMMY)
      GO TO 100
   50 UY=0.0
      VY=0.0
      GO TO 100
   60 DUMMY=DABS(UX)
      UY=DSQRT(DUMMY)
      VY=0.0
  100 RETURN
      END

$ENTRY
```

APPENDIX F

POLYNOMIAL GENERATOR

1. Use of the Programs

Given N linear factors of the form (AX + B) where A and B are complex numbers, the polynomial generator program computes their product resulting in a polynomial of degree N of the form

$$P(X) = a_1X^N + a_2X^{N-1} + \ldots + a_NX + a_{N+1}.$$

Note that if (AX + B) is a factor of the polynomial P(X), then $\frac{-B}{A}$ is a zero of P(X).

Two programs of the polynomial generator are presented here. The first is the single precision program. The second program is in double precision and is designed to perform double precision complex arithmetic. These programs are written for use on any computer using FORTRAN IV language. They have been tested on the IBM S/360 mod. 50 computer which has a 32 bit word.

Each program is designed to construct polynomials of degree 25 or less. In order to consturct polynomials of degree N where N > 25, the array dimensions must be changed as listed in Table XXII for both single precision and double precision programs.

TABLE XXII

ARRAYS USED IN THE POLYNOMIAL GENERATOR

Single Precision		Double Precision
	Main Program	

FACTOR(N,2)
C(N+1)
		FACTOR(N,2),FACTOI(N,2)
		CR(N+1),CI(N+1)

	Subroutine POLY	

FACTOR(N,2)
C(N+1)
D(2)
		FACTOR(N,2),FACTOI(N,2)
		CR(N+1),CI(N+1)
		DR(2),DI(2)

The single precision program may be converted to double precision for use on machines equipped to perform double precision complex arithmetic provided the following changes or their equivalent are made: The changes presented below are those required for the IBM S/360.

1. The format statements should be changed from E-type to D-type.
2. Change COMPLEX $C_1,C_2,\ldots$ to COMPLEX*16 $C_1,C_2,\ldots$ where $C_1,C_2,\ldots$ are complex variables.
3. Add IMPLICIT REAL*8(A-H,O-Z)

After selecting the desired program, the input data should be prepared as described in section 2.

2. Input Data for the Polynomial Generator

The input data for the polynomial generator is grouped into polynomial data sets. Each polynomial data set consists of the data to generate one and only one polynomial. As many polynomials as the user desires may be generated by placing the polynomial data sets one behind the other. Each polynomial data set consists of two kinds of information placed in the following order:

1. Control information
2. Factors of the polynomial

An end card follows the entire collection of data sets. It indicates that there is no more data to follow and terminates execution of the program. This information is displayed in figure 17 and described below. For single precision data, the E-type specification should be used, while for double precision data, the D-type specification should be used. All data should be right justified.

Control Information

The control card is the first card of the polynomial data set and contains the information given in Table XXIII. See figure 14.

TABLE XXIII

CONTROL DATA FOR THE POLYNOMIAL GENERATOR

Variable Name	Card Columns	Description
N	c.c. 1-2	Number of linear factors to be multiplied together. Integer. Right justify.
KCHECK	c.c. 80	This should be left blank.

Factors of the Polynomial

The factors to be read in should be of the form (AX + B) where A and B are non-zero complex numbers. The coefficient of the X term, A, is entered first. Both coefficients of a factor are entered on the same card as described in Table XXIV and illustrated in figure 15. The variable in parentheses is the double precision equivalent. The example in figure 15 is X + (2.5 - 7i).

TABLE XXIV

FACTOR DATA FOR THE POLYNOMIAL GENERATOR

Variable Name	Card Columns	Description
FACTOR(I,1) (FACTOR(I,1))	c.c. 1-20	Real part of the complex number A. Real. Right justify.
FACTOR(I,1) (FACTOI(I,1))	c.c. 21-40	Imaginary part of the complex number A. Real. Right justify.
FACTOR(I,2) (FACTOR(I,2))	c.c. 41-60	Real part of the complex number B. Real. Right justify.
FACTOR(I,2) (FACTOI(I,2))	c.c. 61-80	Imaginary part of the complex number B. Real. Right justify.

End Card

The end card is the last card of the input data to the program.

It indicates that there is no more data to be read. When this card is read, program execution is terminated. This card is described in Table XXV and illustrated in figure 16.

TABLE XXV

DATA TO END EXECUTION OF THE POLYNOMIAL GENERATOR

Variable Name	Card Columns	Description
KCHECK	c.c. 80	Must contain the number 1. Integer.

3. Variables Used in the Polynomial Generator

The definitions of the major variables used in the polynomial generator are given in Table XXVI. The notation and symbols used are described in Appendix B, § 3. Variables not listed are dummy variables used for temporary storage of computations.

4. Description of Program Output

The output from the polynomial generator consists of the following information and is illustrated in Exhibit DD.

The linear factors are printed, in the order read, under the heading "THE XX FACTORS OF THE POLYNOMIAL TO BE GENERATED ARE." XX is the number of linear factors given. The form of each factor is AX + B.

The coefficients of the resulting polynomial are printed, coefficient of highest degree term first, under the heading "THE POLYNOMIAL OF DEGREE XX GENERATED FROM THE ABOVE FACTORS IS." XX is the degree of the polynomial.

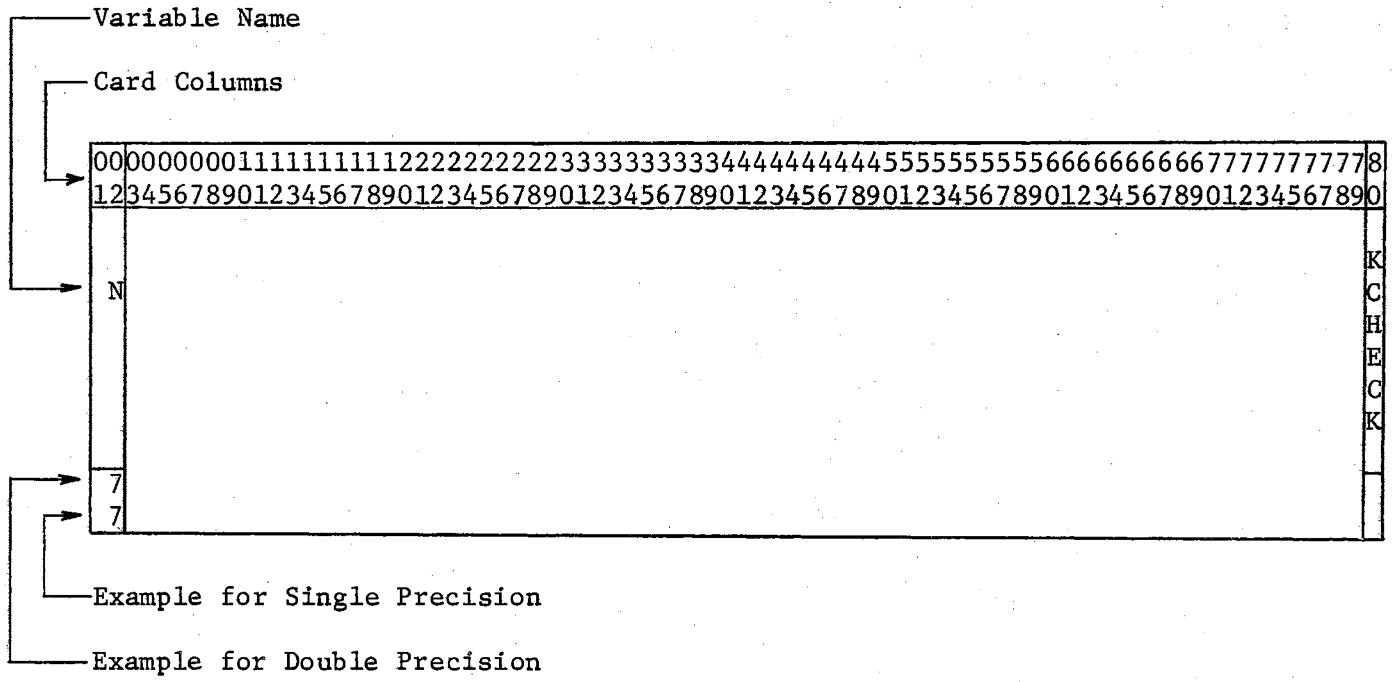


Figure 14. Control Card for the Polynomial Generator

00000000011111111112 12345678901234567890	22222222233333333334 12345678901234567890	44444444455555555556 12345678901234567890	66666666677777777778 12345678901234567890
FACTOR(I,1) (FACTOR(I,1))	FACTOR(I,1) (FACTOI(I,1))	FACTOR(I,2) (FACTOR(I,2))	FACTOR(I,2) (FACTOI(I,2))
0.1D+01 0.1E+01	0.0D+00 0.0E+00	0.25D+01 0.25E+01	-0.7D+01 -0.7E+01

Figure 15. Factor Card for the Polynomial Generator

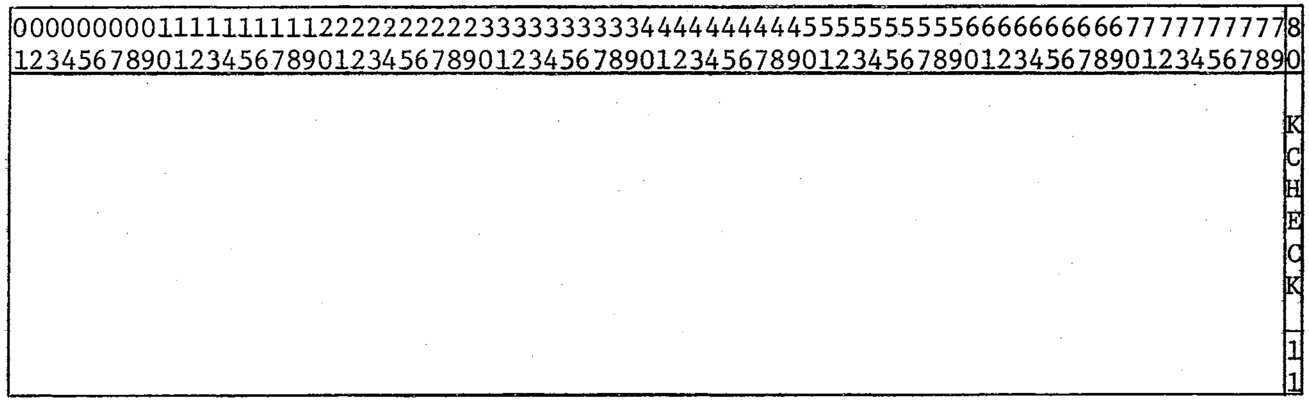


Figure 16. End Card for the Polynomial Generator

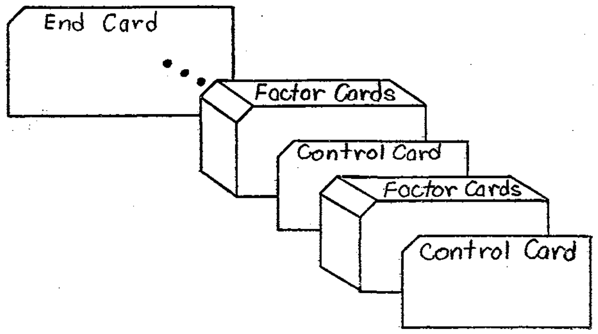


Figure 17. Sequence of Input Data for the Polynomial Generator

TABLE XXVI

VARIABLES USED IN THE POLYNOMIAL GENERATOR

Single Precision Variable	Single Precision Type	Double Precision Variable	Double Precision Type	Disposition of Argument	Description
					Main Program
IO1	I	IO1	I		Unit number of the input device
IO2	I	IO2	I		Unit number of the output device
N	I	N	I		Number of factors to be read
KCHECK	I	KCHECK	I		KCHECK = 1 implies that execution will be terminated
FACTOR	C	FACTOR,FACTOI	R		Array containing the coefficients of the linear factors
C	C	CR,CI	R		Array containing coefficients of the polynomial
					Subroutine POLY
NC	I	NC	I		Counter
C	C	CR,CI	R	R	Array of coefficients of polynomial
FACTOR	C	FACTOR,FACTOI	R	E	Array of factors
N	I	N	I	E	Number of factors to be multiplied

```
THE  7 FACTORS OF THE POLYNOMIAL TO BE GENERATED ARE

FACTOR( 1) =  0.1000000000000000D 01 +  0.0000000000000000D 00 I  X   +   -0.1000000000000000D 01 + -0.1000000000000000D 01 I
FACTOR( 2) =  0.1000000000000000D 01 +  0.0000000000000000D 00 I  X   +   -0.2000000000000000D 01 +  0.3000000000000000D 01 I
FACTOR( 3) =  0.1000000000000000D 01 +  0.0000000000000000D 00 I  X   +    0.1000000000000000 01 +  0.0000000000000000D 00 I
FACTOR( 4) =  0.1000000000000000D 01 +  0.0000000000000000D 00 I  X   +    0.0000000000000000D 00 + -0.2000000000000000 01 I
FACTOR( 5) =  0.1000000000000000D 01 +  0.0000000000000000D 00 I  X   +    0.3000000000000000D 01 + -0.3000000000000000D 01 I
FACTOR( 6) =  0.1000000000000000D 01 +  0.0000000000000000D 00 I  X   +    0.1000000000000000D 01 +  0.1000000000000000 01 I
FACTOR( 7) =  0.1000000000000000D 01 +  0.0000000000000000D 00 I  X   +    0.1000000000000000D 01 +  0.1000000000000000D 01 I


THE POLYNOMIAL OF DEGREE  7 GENERATED FROM THE ABOVE FACTORS IS


C( 1) =  0.1000000000000000D 01 +  0.0000000000000000D 00 I
C( 2) =  0.3000000000000000D 01 + -0.1000000000000000D 01 I
C( 3) =  0.8000000000000000D 01 +  0.9000000000000002D 01 I
C( 4) =  0.2399999999999999D 02 +  0.1600000000000000D 02 I
C( 5) =  0.7799999999999998D 02 +  0.1800000000000000D 02 I
C( 6) =  0.7999999999999998D 02 + -0.2800000000000001D 02 I
C( 7) =  0.6799999999999999D 02 + -0.1120000000000000D 03 I
C( 8) =  0.4800000000000000D 02 + -0.7200000000000001D 02 I

COMPILE TIME=     1.96 SEC,EXECUTION TIME=     0.49 SEC,OBJECT CODE=    3472 BYTES,ARRAY AREA=     1248 BYTES,UNUSED=    65280 BYTES

COMPILE TIME=     0.11 SEC,EXECUTION TIME=     0.00 SEC,OBJECT CODE=    3472 BYTES,ARRAY AREA=     1248 BYTES,UNUSED=    65280 BYTES
          $STOP
```

Exhibit DD. Linear Factors Are: X + (-1 - i), X + (-2 + 3i), X + 1, X + (-2i), X + (3 - 3i), X + (1 + i), X + (1 + i).

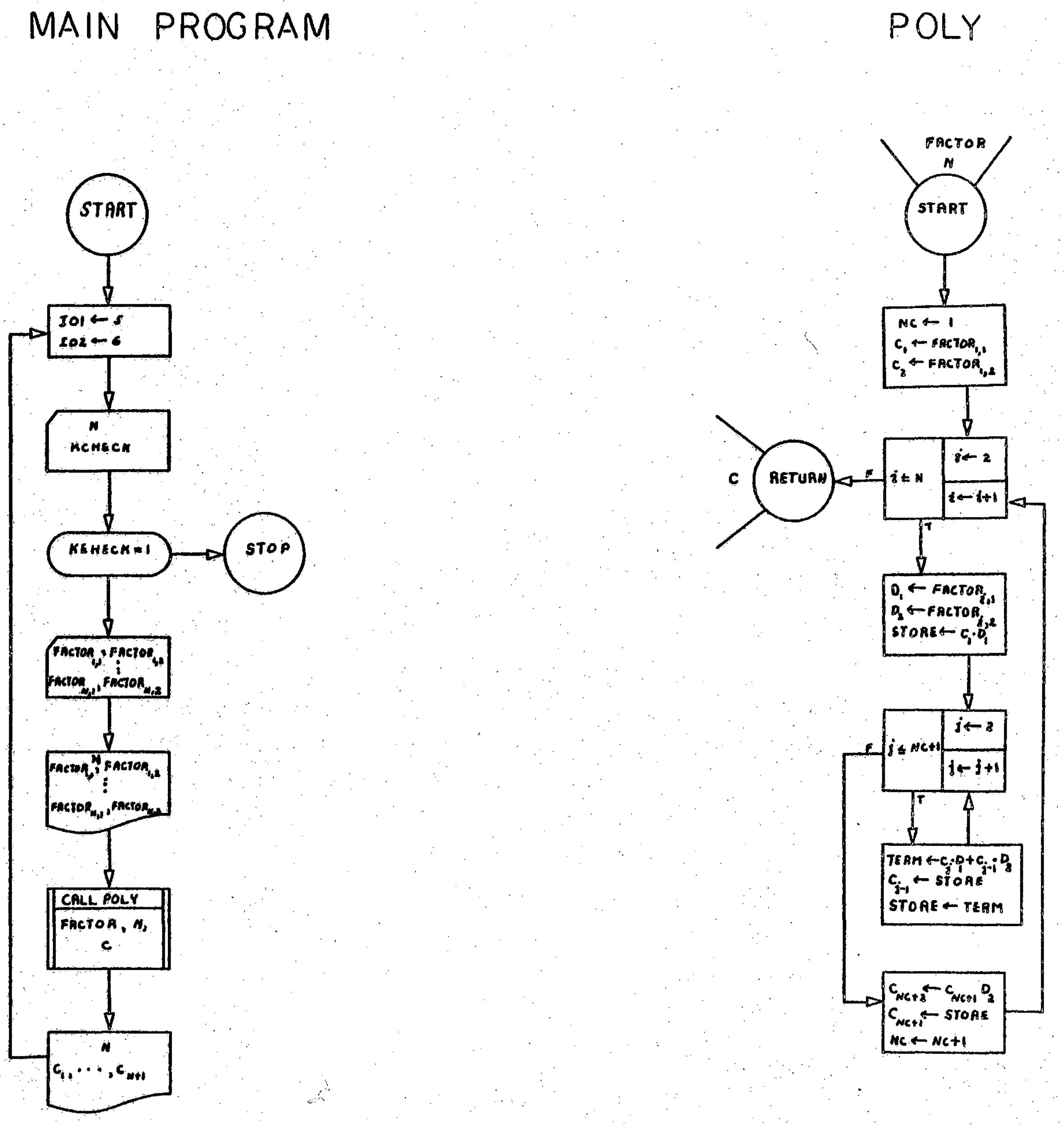


Figure 18. Flow Charts for the Polynomial Generator

TABLE XXVII

SINGLE PRECISION PROGRAM FOR THE POLYNOMIAL GENERATOR

```
$JOB 10414
C     ***********************************************************************
C     *                                                                     *
C     * SINGLE PRECISION PROGRAM FOR THE POLYNOMIAL GENERATOR.              *
C     *                                                                     *
C     *                                                                     *
C     * THIS PROGRAM CONSTRUCTS THE N-TH DEGREE POLYNOMIAL FORMED BY        *
C     * MULTIPLYING N GIVEN LINEAR FACTORS TOGETHER.                        *
C     *                                                                     *
C     ***********************************************************************
      COMPLEX FACTOR,C
      DIMENSION FACTOR(25,2),C(26)
      IO1=5
      IO2=6
    1 READ(IO1,1000) N,KCHECK
      IF(KCHECK.EQ.1) STOP
      READ(IO1,1010) (FACTOR(I,1),FACTOR(I,2),I=1,N)
      WRITE(IO2,1020) N
      WRITE(IO2,1030) (I,FACTOR(I,1),FACTOR(I,2),I=1,N)
      CALL POLY(FACTOR,N,C)
      WRITE(IO2,1040) N
      KKK=N+1
      WRITE(IO2,1050) (I,C(I),I=1,KKK)
      GO TO 1
 1000 FORMAT(I2,77X,I1)
 1010 FORMAT(4E20.0)
 1020 FORMAT(1H1,4HTHE ,I2,46H FACTORS OF THE POLYNOMIAL TO BE GENERATED
     1 ARE//)
 1030 FORMAT(1X,7HFACTOR(,I2,4H) = ,E14.7,3H + ,E14.7,5H I  X,7H    +   ,
     1E14.7,3H + ,E14.7,2H I)
 1040 FORMAT(///1X,25HTHE POLYNOMIAL OF DEGREE ,I2,36H GENERATED FROM TH
     1E ABOVE FACTORS IS//)
 1050 FORMAT(1X,2HC(,I2,4H) = ,E14.7,3H + ,E14.7,2H I)
      END


      SUBROUTINE POLY(FACTOR,N,C)
C     ***********************************************************************
C     *                                                                     *
C     * GIVEN N LINEAR FACTORS OF THE FORM (AX+B) WHERE A AND B ARE COMPLEX, *
C     * SUBROUTINE POLY FORMS THEIR PRODUCT.                                *
C     *                                                                     *
C     ***********************************************************************
      COMPLEX FACTOR,C,D,STORE,TERM
      DIMENSION FACTOR(25,2),C(26),D(2)
      NC=1
      C(1)=FACTOR(1,1)
      C(2)=FACTOR(1,2)
      DO 50 I=2,N
      D(1)=FACTOR(I,1)
      D(2)=FACTOR(I,2)
      STORE=C(1)*D(1)
      KKK=NC+1
      DO 20 J=2,KKK
      TERM=(C(J)*D(1))+(C(J-1)*D(2))
      C(J-1)=STORE
   20 STORE=TERM
      C(NC+2)=C(NC+1)*D(2)
      C(NC+1)=STORE
   50 NC=NC+1
      RETURN
      END
```

TABLE XXVIII

DOUBLE PRECISION PROGRAM FOR THE POLYNOMIAL GENERATOR

```
$JOB 10414
C     ************************************************************************
C     *                                                                      *
C     * DOUBLE PRECISION PROGRAM FOR THE POLYNOMIAL GENERATOR.               *
C     *                                                                      *
C     *                                                                      *
C     * THIS PROGRAM CONSTRUCTS THE N-TH DEGREE POLYNOMIAL FORMED BY         *
C     * MULTIPLYING N GIVEN LINEAR FACTORS TOGETHER.                         *
C     *                                                                      *
C     ************************************************************************
      DOUBLE PRECISION FACTOR,FACTOI,CR,CI
      DIMENSION FACTOR(25,2),FACTOI(25,2),CR(26),CI(26)
      IO1=5
      IO2=6
    1 READ(IO1,1000) N,KCHECK
      IF(KCHECK.EQ.1) STOP
      READ(IO1,1010) (FACTOR(I,1),FACTOI(I,1),FACTOR(I,2),FACTOI(I,2),I=
     11,N)
      WRITE(IO2,1020) N
      WRITE(IO2,1030) (I,FACTOR(I,1),FACTOI(I,1),FACTOR(I,2),FACTOI(I,2)
     1,I=1,N)
      CALL POLY(FACTOR,FACTOI,N,CR,CI)
      WRITE(IO2,1040) N
      KKK=N+1
      WRITE(IO2,1050) (I,CR(I),CI(I),I=1,KKK)
      GO TO 1
 1000 FORMAT(I2,77X,I1)
 1010 FORMAT(4D20.0)
 1020 FORMAT(1H1,4HTHE ,I2,46H FACTORS OF THE POLYNOMIAL TO BE GENERATED
     1 ARE//)
 1030 FORMAT(1X,7HFACTOR(,I2,4H) = ,D23.16,3H + ,D23.16,5H I  X,7H   +
     1 ,D23.16,3H + ,D23.16,2H I)
 1040 FORMAT(///1X,25HTHE POLYNOMIAL OF DEGREE ,I2,36H GENERATED FROM TH
     1E ABOVE FACTORS IS//)
 1050 FORMAT(1X,2HC(,I2,4H) = ,D23.16,3H + ,D23.16,2H I)
      END
```

TABLE XXVIII (Continued)

```
      SUBROUTINE POLY(FACTOR,FACTOI,N,CR,CI)
C    ************************************************************************
C    *                                                                      *
C    * GIVEN N LINEAR FACTORS OF THE FORM (AX+B) WHERE A AND B ARE COMPLEX, *
C    * SUBROUTINE POLY FORMS THEIR PRODUCT.                                 *
C    *                                                                      *
C    ************************************************************************
      DOUBLE PRECISION FACTOR,FACTOI,CR,CI,STORER,STOREI,TERMR,TERMI,DR,
     1DI
      DIMENSION FACTOR(25,2),FACTOI(25,2),CR(26),CI(26),DR(2),DI(2)
      NC=1
      CR(1)=FACTOR(1,1)
      CI(1)=FACTOI(1,1)
      CR(2)=FACTOR(1,2)
      CI(2)=FACTOI(1,2)
      DO 50 I=2,N
      DR(1)=FACTOR(I,1)
      DI(1)=FACTOI(I,1)
      DR(2)=FACTOR(I,2)
      DI(2)=FACTOI(I,2)
      STORER=CR(1)*DR(1)-CI(1)*DI(1)
      STOREI=CI(1)*DR(1)+CR(1)*DI(1)
      KKK=NC+1
      DO 20 J=2,KKK
      TERMR=(CR(J)*DR(1)-CI(J)*DI(1))+(CR(J-1)*DR(2)-CI(J-1)*DI(2))
      TERMI=(CI(J)*DR(1)+CR(J)*DI(1))+(CI(J-1)*DR(2)+CR(J-1)*DI(2))
      CR(J-1)=STORER
      CI(J-1)=STOREI
      STORER=TERMR
   20 STOREI=TERMI
      CR(NC+2)=CR(NC+1)*DR(2)-CI(NC+1)*DI(2)
      CI(NC+2)=CI(NC+1)*DR(2)+CR(NC+1)*DI(2)
      CR(NC+1)=STORER
      CI(NC+1)=STOREI
   50 NC=NC+1
      RETURN
      END

$ENTRY
```

VITA

2

Randy Joe Snider

Candidate for the Degree of

Master of Science

Thesis: NUMERICALLY APPROXIMATING ZEROS OF A POLYNOMIAL

Major Field: Mathematics

Biographical:

Personal Data: Born in Oklahoma City, Oklahoma, August 16, 1945, the son of Bill M. and Dorothy Elizabeth Snider.

Education: Graduated from Antlers High School, Antlers, Oklahoma, in May, 1963; received the Bachelor of Science degree with a major in mathematics from Oklahoma State University in 1968; completed requirements for Master of Science degree at Oklahoma State University in May, 1970.

Professional Experience: Graduate Assistant, Mathematics Department, Oklahoma State University, 1968 and 1969; Programmer, Skelly Oil Company, summer 1968; has accepted an appointment as Systems Analyst for Skelly Oil Company, Tulsa, Oklahoma.

Organizations: Member of Triangle Fraternity, national, social, professional fraternity of engineers, architects, and scientists.